Verteilte Systeme

Ausarbeitung des Fragenkataloges der Vorlesung v2

WS 08/09

1)
Geben Sie eine charakterisierende Definition für ein "Verteiltes System" an. Nennen Sie die wichtigsten Design-Ziele bzw. charakteristischen Eigenschaften von Verteilten Systemen. Stellen Sie weiters den Zusammenhang zu den typischen Fallstricken beim Entwurf verteilter Systeme her.

Definition: Ein verteiltes System ist eine Menge voneinander unabhängiger Computer, die dem Benutzer wie ein einzelnen, kohärentes System erscheinen.

Design-Ziele:

- Um heterogene Computer und Netzwerke zu unterstützen und gleichzeitig das Erscheinungsbild eines einzigen Systems anzubieten, werden verteilte Systeme häufig mit Hilfe einer Softwareschicht angeordnet.
- Ein solches verteiltes System wird als Middleware bezeichnet.
- Verteilte Systeme unterscheiden sich insofern von herkömmlicher Software, da ihre Komponenten über ein Netz verteilt sind.
- Beim Entwurf muss die Verteilung berücksichtigt werden, da ansonsten die Systeme unnötig zu komplex werden.
- Transparenz
- Erweiterbarkeit
- Fehlertoleranz
- Gleichzeitiges Nutzen von Ressourcen (Resource Sharing)

Sechs charakteristische Eigenschaften:

- Ein verteiltes System besteht aus autonomen Komponenten.
- Die Benutzer (Menschen oder Programme) glaube, sie hätten es mit einem einzigen System zu tun.
- Unterschiede zwischen den einzelnen Computern und der Art wie sie miteinander kommunizieren, bleiben weitgehend vor den Benutzern verborgen. Ebenso der innere Aufbau.
- Benutzer können konsistent und einheitlich mit dem System arbeiten
- Grundsätzlich sollten verteilte Systeme einfach zu erweitern oder zu skalieren sein.
- Es steht normalerweise ununterbrochen zur Verfügung, auch wenn einige Komponenten vorübergehend ausfallen.

Typische Fallstricken

- Das Netzwerk ist zuverlässig
- Das Netzwerke ist sicher
- Das Netzwerk ist homogen
- Die Topologie ändert sich nicht
- Die Latenzzeit beträgt 0.

- Die Bandbreite ist unbegrenzt
- Die Übertragungskosten betragen 0.
- Es gibt genau einen Administrator.

2)
Was ist Transparenz? Beschreiben Sie die standardisierten Arten von Transparenz und erklären Sie den Zusammenhang zischen den einzelnen Transparenz-Definitionen. Was ist der Nachteil von Transparenz?

Transparenz bedeutet:

Die Verteilung des Systems soll vom Benutzer verborgen werden.

6 Arten von Transparenz:

- **ORTSTRANSPARENZ**
  die geographische Position soll verborgen werden
- **FEHLERTRANSPARENZ**
  Fehler werden verborgen. Verbirgt den Ausfall und die Wiederherstellung einer Ressource.
- **ZUGRIFFSTRANSPARENZ**
  Verbirgt Unterschiede in der Datendarstellung und die Art und Weise, wie auf eine Ressource zugegriffen wird. Unterschiede in Computerarchitektur, wie Daten von verschiedenen Computern und Betriebssystemen dargestellt werden (Namenskonventionen etc.)
- **NEBENLÄUFIGKEITSTRANSPARENZ**
  Verbirgt, dass eine Ressource von mehreren konkurrierenden Benutzern gleichzeitig genutzt werden kann.
- **MIGRATIONSTRANSPARENZ**
  Verbirgt, dass eine Ressource an einen anderen Ort verschoben werden kann (ohne dass sich an der Art des Zugriffs etwas ändert).
- **RELOKATIONSTRANSPARENZ**
  Verbirgt, dass eine Ressource repliziert ist, also dass mehrere Kopien einer Ressource vorhanden sind (notwendig, dass alle Replikas denselben Namen tragen -> System sollten daher auch Ortstransparenz unterstützen).

Nachteil von Transparenz:

Nicht für alle Situationen ist es sinnvoll, alle Aspekte der Verteiltheit vom Benutzer zu verstecken.

Nicht für alle Situationen kann eine Transparenz garantiert werden, zB bei Update-Operationen auf entfernten Komponenten wird aufgrund der Latenz die Verteilung sichtbar.

Extrembeispiel Drucker – es ist sinnvoll, den Standort des Druckers zu kennen (ORTSTRANSPARENZ ist hier nicht sinnvoll)

Vollständige Verteilungstransparenz ist oft nicht möglich, daher Transparenz vernünftig einsetzen, um das eine oder andere Verhalten des Systems verständlicher zu machen.

3)
Was versteht man unter „OPENNESS“ ?

Ein offenes verteiltes System bietet Dienst nach Standardregeln an, die die Syntax und die Semantik dieser Dienste beschreiben.

Bei Openness geht es vor allem um standardisierte Schnittstellen, die das System zur Verfügung stellt.

Standards in

- Syntax
- Semantik
- Formalisierung in Protokollen
- Flexibilität -> Zusammenstellen, Konfigurationen und Austausch von Komponenten, Erweiterbarkeit

Um Flexibilität in einem VS zu erreichen, sollte aus möglichst kleinen einfachen Komponenten bestehen, die einfach anzupassen und austauschbar sind. Klare Schnittstellen sowohl zu den Anwendungen als auch für die interne Zusammenarbeit.

Es ist eine Trennung zwischen Richtlinie (Policy) und Mechanismus notwendig. Besagt im wesentlichen, dass man mit Parametern den Mechanismus steuern kann. Beispielsweise das Caching-Verhalten eines Webbrowsers.

Maßzahlen für Openness:

VOLLSTÄNDIGKEIT

Beschreibt das Ausmaß, wie das System mit Komponenten anderer Hersteller zusammenarbeiten kann -> INTEROPERABILITÄT

NEUTRALITÄT

Beschreibt das Ausmaß, dass eine Komponente die für System A entwickelt wurde auch auf System B ohne Veränderungen läuft, das die gleichen Schnittstellen implementiert (Austauschbarkeit).

4)
Erläutern Sie die Probleme und Lösungsansätze für „Scalability“.

Probleme:

- Es werden mehr User und Resourcen in das System integriert, als ursprünglich geplant, was eine höhere Bandbreite erfordert -> das Netz wird überlastet
- Bei geographischer Ausdehnung des Systems wird die Latenzzeit höher, sodass eine sofortige synchrone Kommunikation nicht mehr möglich ist.
- Es werden mehr Organisationen/Firmen eingebunden, jedoch widersprechen sich die Schnittstellen (Policies).

Lösungsansätze:

- Replikation -> Vervielfältigung von Ressourcen/Elementen um die Last zu verteilen (Performanceprobleme) zB über Content Delivery Networks
- Verwendung von asynchronen Methoden um Latenzzeiten zu verbergen -> zB asynchrones RPC – Methode werden aufgerufen, und dann sofort mit der Arbeit weitergemacht - > sinnvoll wenn die Antwort nicht unmittelbar erwartet wird (zB eine entfernte UPDATE-Operation).
- Caching als Spezialform von Replikation: Daten werden am Client zwischen gespeichert zB.: die Caches von Web-Browsern.

Probleme bei der Skalierbarkeit:

kann im wesentlichen in den folgenden 3 Dimensionen gemessen werden:

- GRÖSSE
  Werden mehrere Benutzer oder Ressourcen unterstützt, wird man häufig mit den Einschränkungen zentralisierter Dienste, Daten und Algorithmen konfrontiert.

  - Zentraler Services (zB ein einzelner Server für alle Benutzer)
  - Zentrale Daten (zB ein einziges Online-Telefonbuch-Datenbank)
  - Zentrale Algorithmen, sollten verwendet werden.

- GEOGRAPHISCH
  Verteilte Systeme die für LANs entwickelt wurden sind bei geographischer Ausdehnung problematisch wegen zu langer Latenzzeiten .

- ADMINISTRATIV
  Verteilte Systeme über mehrerer unabhängige Domänen sind schwierig, da es unterschiedliche Ansätze zur Benutzung von Ressourcen, Verwaltung und Sicherheit gibt.

Skalierungstechniken (Lösungsansätze):

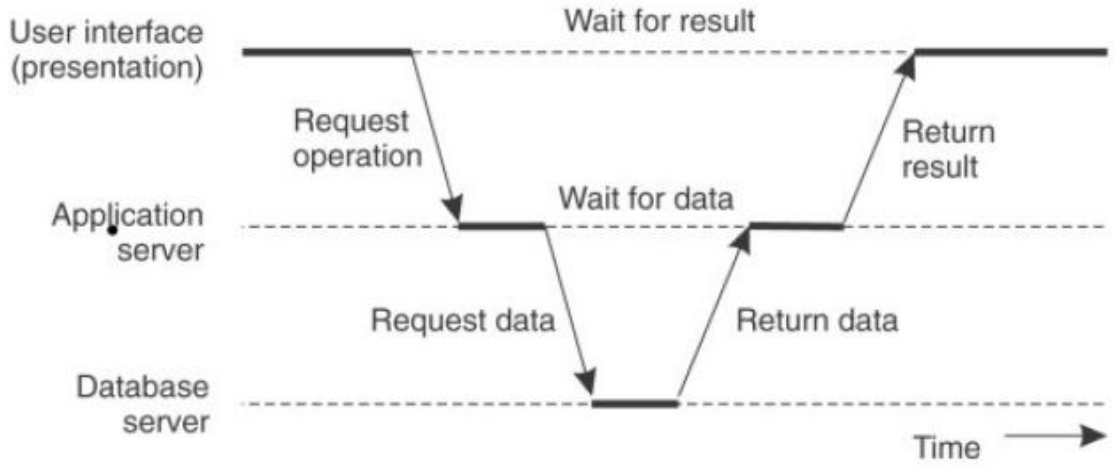
- Verbergen das Latenzzeiten (asynchrone Kommunikation, teile auf Client-Prozess verschieben -> Reduktion der Kommunikation)
- Verteilung (Komponenten in kleinere Teile zerlegen -> zB DNS – Server für jede TOP-Level-Domain).
- Replikation
  erhöht Verfügbarkeit, bei WANs näher gelegene Komponenten verwenden

5)
Was versteht man unter der vertikalen Verteilung bzw. N-Schichten-Systemen? Diskutieren Sie dabei alle Grundvarianten von Client/Server-Systemen. Ist folglich ein Java Applet eher ein Thick Client oder ein Thin Client?

Vertikale Verteilung bzw. N-Schichten System:

Logisch unterschiedliche Komponenten werden auf (**PHYSISCH)** unterschiedlichen Maschinen angeordnet.

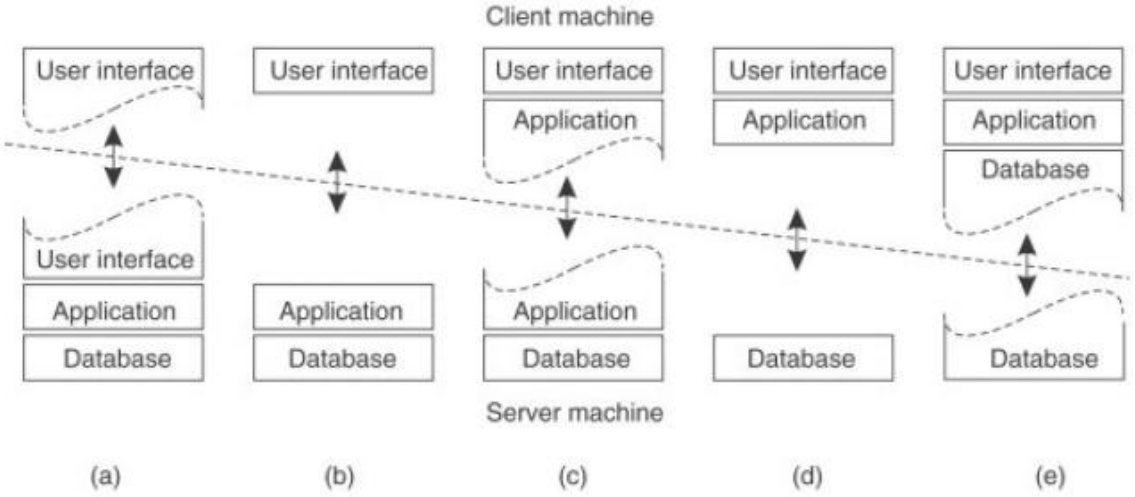
Das ist eine MULTI-TIER-ARCHITEKTUR oder Client/Server Modell.



Umfasst in der Regel mindestens 3 Schichten:

- Benutzeroberfläche
- Verarbeitungsebene (Business Logic) -> Kernfunktionalität
- Datenebene (unabhängige Datenspeicherung in Datenbank)

Ein JAVA-APPLET ist eher ein THICK-Client, da die meiste Funktionalität sich am Client befindet (bei d).



6)
Was ist horizontale Verteilung? Mit welchen grundlegenden Design-Fragen müssen Sie sich beim Entwurf der horizontalen Verteilung eines Systems beschäftigen? Gibt es einen Zusammenhang zur vertikalen Verteilung?

Horizontale Verteilung dient zur Verteilung der Last (Replikation). So werden logisch gleichwertige Teile auf physisch unterschiedlichen Systemen verteilt. Jeder teil arbeitet aber mit einem Anteil der Datenmenge, sodass die Gesamtlast gleichmäßig verteilt wird.

Design-Fragen:

Auswahl eines geeigneten Kosistenzmodelles um sicherzustellen, dass alle Teile/Server mit den gleichen/richtigen Datenteilen arbeiten.

DATENZENTRIERTE KONSISTENZMODELLE

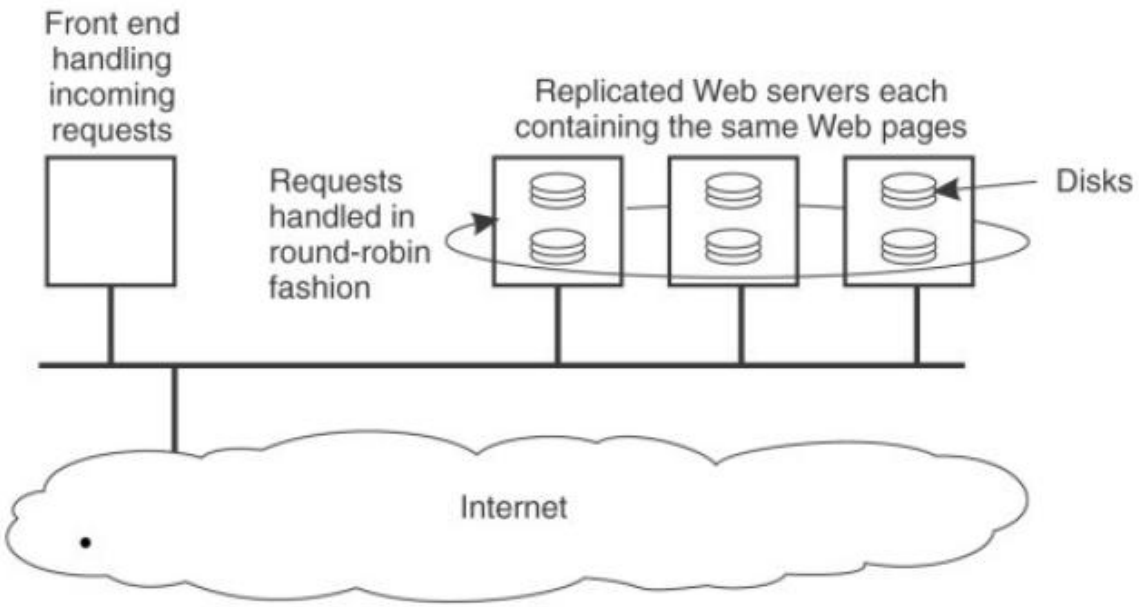
- Strenge
- Sequentielle
- Kausale-Konsistenz
- FIFO-Konsistenz

CLIENTZENTRIERTE KONSISTENZMODELLE

- Eventuelle Konsistenz
- Monotones Lesen

Vertikale Verteilung kann selbstverständlich mit einer horizontalen Kombiniert werden:

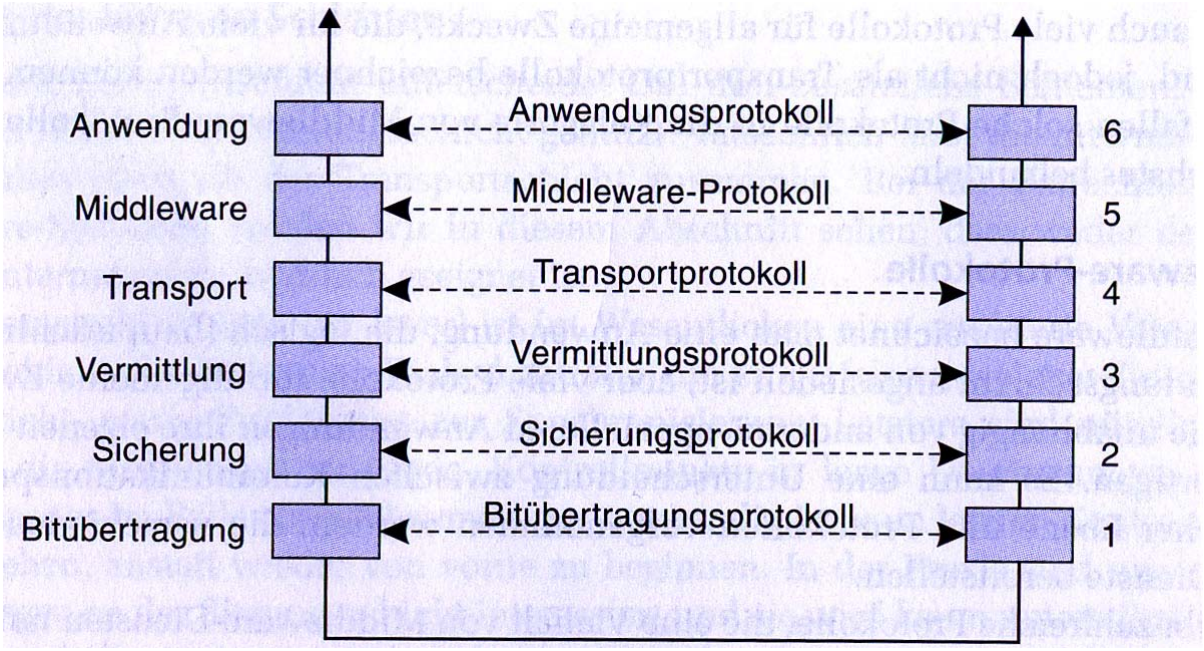
Beispiel Webserver:



Jeder Server verwaltet dieselbe Menge an Webseiten, und immer wenn eine Webseite aktualisiert wird, wird die aktualisierte Version auf alle Server kopiert. (je nach Konsistenzmodell verschieden schnell).

Bekannte Vertreter von horizontalen Verteilungen sind PEER-to-PEER-Systeme. Jeder Prozess agiert gleichzeitig als Client und Server.

7)
Beschreiben Sie das ISO-OSI Modell der geschichteten Protokolle (Grundprinzip). Stellen Sie den Bezug zu den Internet-Protokollen (TCP/IP) her. Warum sind Transport-Layer Protokolle für Verteilte Systeme oft nicht ausreichend?



Ist die Grundlage für die Kommunikation in verteilten Systemen. Beim ISO/OSI-Modell werden eine Reihe von Vereinbarungen bei der Übertragung getroffen und diese Aufgaben in verschiedene Schichten unterteilt, um Chaos zu vermeiden. Es ermöglicht die Kommunikation in offenen Systemen.

Diese Kommunikationsregeln wird als Protokoll bezeichnet.

Im OSI-Modell wird die Kommunikation in 7 Schichten aufgeteilt. Jede Schicht behandelt einen anderen Aspekt. Damit kann die Kommunikation in kleinere, handliche Teile zerlegt werden.

Bei Übermittlung (Senden) einer Nachricht übergibt die Anwendungsschicht die Nachricht an die untere Ebene und fügt einen Header hinzu. Diese gibt die Nachricht wiederum weiter bis nach unten. Auf der Physischen Schicht wird dann die Nachricht an das Netzwerk übergeben (Bits).

PHYSICAL LAYER (Bitübertragung)

Die Bitübertragungsschicht ist für das Übertragen der Einsen und Nullen zuständig. Sie kümmert sich um die Standardisierung, sodass von einem Rechner gesendete 1 auch tatsächlich als 1 und nicht als 0 gesendet wird. Hierbei können allerdings Fehler auftreten, sodass ein Verfahren benötigt wird, um diese zu entdecken und zu korrigieren.

DATA LINK LAYER (Sicherungsschicht)

Sie gruppiert Bits in Frames und berechnet eine Prüfsumme. Der Empfänger berechnet die Prüfsumme erneut. Stimmen beide überein, so wird der Frame als korrekt anerkannt, wenn nicht bittet der Empfänger den Sender um erneute Übertragung.

NETWORK LAYER (Vermittlung)

Routing-Aufgabe. Sicht den besten Weg im Netzwerk. Die Auswahl des besten Wegs (beste Teilstrecken) ist die Aufgabe des NETWORK-LAYERs.

TRANSPORT LAYER (Transport)

Die Transportschicht macht das zugrundeliegende Netzwerk dem Anwendungsentwickler nutzbar. Die Idee ist, dass eine Anwendung eine Nachricht an die Tansportschicht übermittelt mit der Erwartung, dass diese sicher ihr Ziel erreicht. Sie zerteilt eine Nachricht in Stücke, die klein genug sind für eine Übertragung. Es kann passieren, dass beim Empfangen die Pakete nicht in der richtigen Reihenfolge eintreffen -> die Transportschicht muss dies ausbügeln.

Verbindungsorientiert: TCP
Verbindungslos: UDP
Echtzeitübertragung: RTP

SESSION LAYER (Sitzungsschicht)

Ist im wesentliche eine erweiterte Transportschicht, die in der Internetprotokollsuite nicht vorhanden ist. In der Praxis wenig relevant, bietet Funktionen zur Dialogkontrolle (Teilnehmer) und Synchronisierung.

APPLICATION LAYER (Anwendung)

Netzwerkanwendungen:

- FTP-Protokoll (Zur Übertragung von Dateien)
- HTTP-Protokoll (Zur Übermittlung von Webseiten)

Im OSI fehlt eine klare Unterscheidung zwischen Anwendungen und anwendungsspezifischen Protokollen.

8)
Was ist Middleware? Welche Anforderungen stellt man an Middleware? Welche Services soll Middleware bieten? Erläutern Sie den Zusammenhang zwischen Middleware und Architectural styles.

Was ist Middleware?

Middleware bildet eine Schicht zwischen Anwendungen und verteilten Plattformen. Ein wichtiger Zweck besteht darin, einen Grad von Verteilungstransparenz zu bieten, er in einem gewissen Maße die Verteilung von Daten verbirgt. Sie fasst gewisse Kernfunktionalität zusammen und bietet diese über Services im System an. Sie ist logisch zwischen Applikation und Transportlayer einzuordnen.

Beispiele für Middleware:

- Verschlüsselung
- Authentifizierung
- Security
- Replikation
- Access-Transparenz
- Naming

Die Entwicklung von Middleware beschäftigt sich mit der Entwicklung von zusätzlicher Funktionalität für Anwendungen. Unabhängige Dienste für verschiedene Anwendungen.

Welche Anforderungen stellt man an Middleware?

- Flexibilität
- Abstraktion
- Transparen**z**

Welche Services soll Middlware bieten?

- Kommunikationseinrichtungen für Zugriffstransparenz
- Namensgebung für Orts- und Migrationstransparenz
- Distributed transactions for concurrency transparency; distributed locking
- Verteilte Transaktionen für Nebenläufigkeitstransparenz, verteiltes Locking
- Replikation zur unterstützung für Fehlertransparenz
- Security Authentication, authorization
- Sicherheitsauthentifikation, Autorisation

➔ mainly concern extra-functional (quality) properties

Erläutern Sie den Zusammenhang zwischen Middleware und Architectural styles:

Bei Middleware kommen verschiedene softwarearchitektonische Stile zum Einsatz:

- Geschichtete Architekturen (OSI)
- Objektbasierte Architekturen und Komponenten
- Datenzentrierte Architekturen (Dateibasiert, Datenbank)
- Ereignisbasierte Architekturen

9)
Wie kann man die Flexibilität der Middleware erhöhen sowie die Zusammenarbeit von Middleware und Anwendung effizienter gestalten? Erläutern Sie dabei die Grundprinzipien von Interceptoren, Adaptivität und Self-Management.

Wie kann man die Flexibilität der Middleware erhöhen sowie die Zusammenarbeit von Middleware und Anwendung effizienter gestalten?

Einsatz von Architectural styles.

Grundprinzipien:

<u>Interceptoren</u>

- Unterbricht den gewöhnlichen Programmfluss.
- Erlaubt die Ausführung anderem anwendungsspezifischen Code auszuführen
- Für generische Interzeptoren muss möglicherweise ein höherer Implementierungsaufwand erbracht werden.

<u>Adaptivität (allgemeine Ansätze für adaptive Software)</u>

Es werden drei grundlegende Techniken unterschieden:

- TRENNEN DER BELANGE (Separation of concerns)
  -> herkömmliche Modularisierung von Systemen
- REFLEKTION
  -> Bezieht sich auf die Möglichkeit eines Programms, sich selbst zu untersuchen und gegebenfalls bei Bedarf sein Verhalten anzupassen.
  -> Wurde in Java eingebaut.
- KOMPONENTENBASIERTES DESIGN
  -> Ein System kann entweder statisch oder dynamisch zur Laufzeit konfiguriert werden. Module laden- und entladen.

Self-Management:

Bei vereilten Systemen ist nicht immer eine zentrale Verwaltung/Organisation eine optimale Lösung.

Bei Selbstmanagement in verteilten Systemen betrachten wir **rückkopplungsgesteuerte Systeme,** die eine automatische Anpassung an Änderungen ermöglichen.

- Selbstverwaltend
- Selbstheilend
- Selbstkonfigurationierend
- Selbstoptimierend usw.

10)
Erläutern Sie das Grundprinzip des Remote Procedure Call. Gehen Sie auf die Begriffe "client stub" und "server stub" näher ein.

RPC
Programme erhalten die Möglichkeit Prozeduren auf entfernten Rechnern auszuführen.

Wenn ein Prozess auf Rechner A eine Prozedur auf dem Rechner B aufruft, wird der Prozess auf A angehalten und die Ausführung der aufgerufenen Prozedur findet auf Rechner B statt.

Es können auch Ergebnisse von der entfernt aufgerufenen Methode zurückgegeben werden.

CLIENT STUB

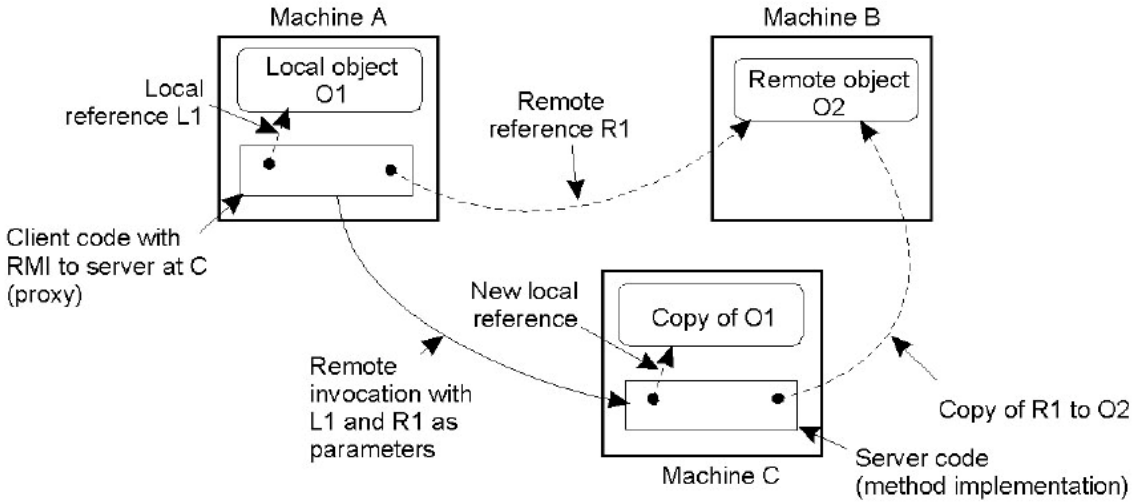
Kapselt den entfernen Aufruf. Das Programm sieht den Aufruf als lokal an. Der Client-Stub übernimmt die Weiterleitung des Aufrufs an den Server.

SERVER STUB

Kapsel die eigentliche Prozedur, die aufgerufen wird. Er nimmt den Aufruf des Client-Stub entgegen. Für den Server sieht der Aufruf so aus, als käme er von einem lokalen Programm.

11)
Wie können Variablen bei Prozedur-Aufrufen grundsätzlich übergeben werden? Wie werden Sie bei RPC gehandhabt und welche Probleme gibt es dabei? Was versteht man in diesem Zusammenhang unter "parameter marshalling"?

Wie können Variablen bei Prozedur-Aufrufen grundsätzlich übergeben werden?



Die Übergabe erfolgt entweder als Referenz oder als Wert. Für die aufgerufene Prozedur ist ein Wertparameter lediglich eine initialisierte lokale Variable. Sie kann ihn ändern, aber solche Änderungen wirken sich nicht auf den Originalwert auf der Seite des Aufrufers aus. Bei Übergabe einer Referenz, wird bei Änderung der Originalwert verändert (keine Kopie).

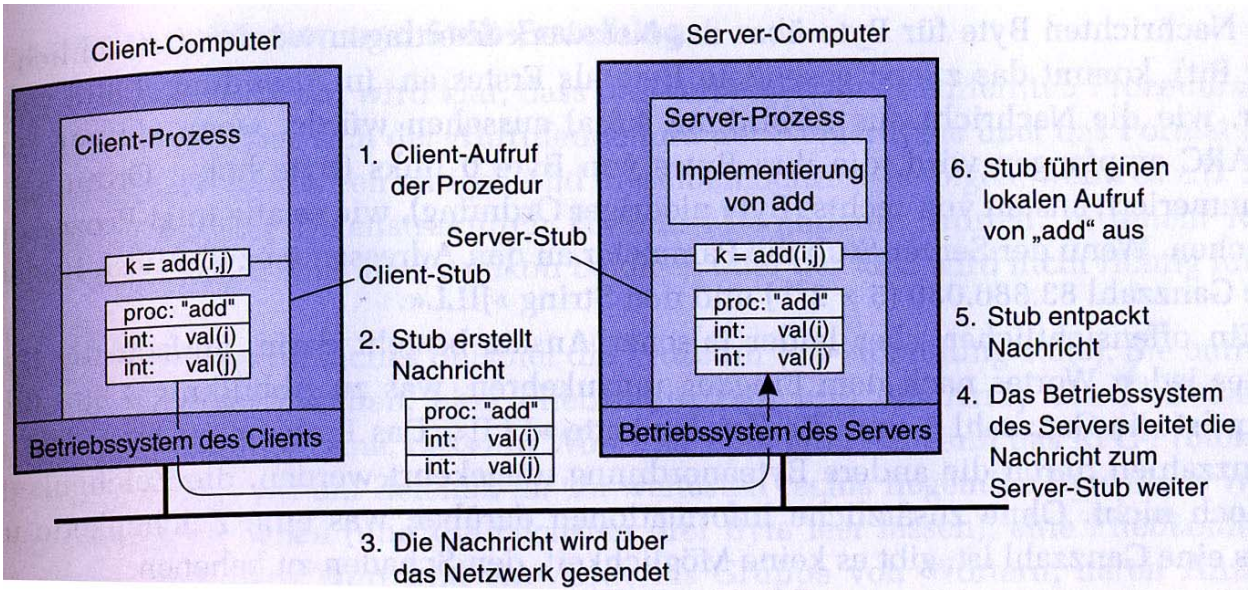
Wie werden Sie bei RPC gehandhabt und welche Probleme gibt es dabei?

Bei RPC erfolgt die Übergabe grundsätzlich als Wert, da eine Übergabe als Referenz nicht sinnvoll ist. Bei Übergabe von Referenzen wird dies in Copy/Paste aufgelöst -> dh es wird zuerst die Referenz aufgelöst und der konkrete Wert übergeben.

Was versteht man in diesem Zusammenhang unter "parameter marshalling"?

Das Verpacken von Parametern in eine Nachricht wird Parameter-Marshaling genannt.
Diese Nachrichten werden dann an den Server-Stub gesendet.

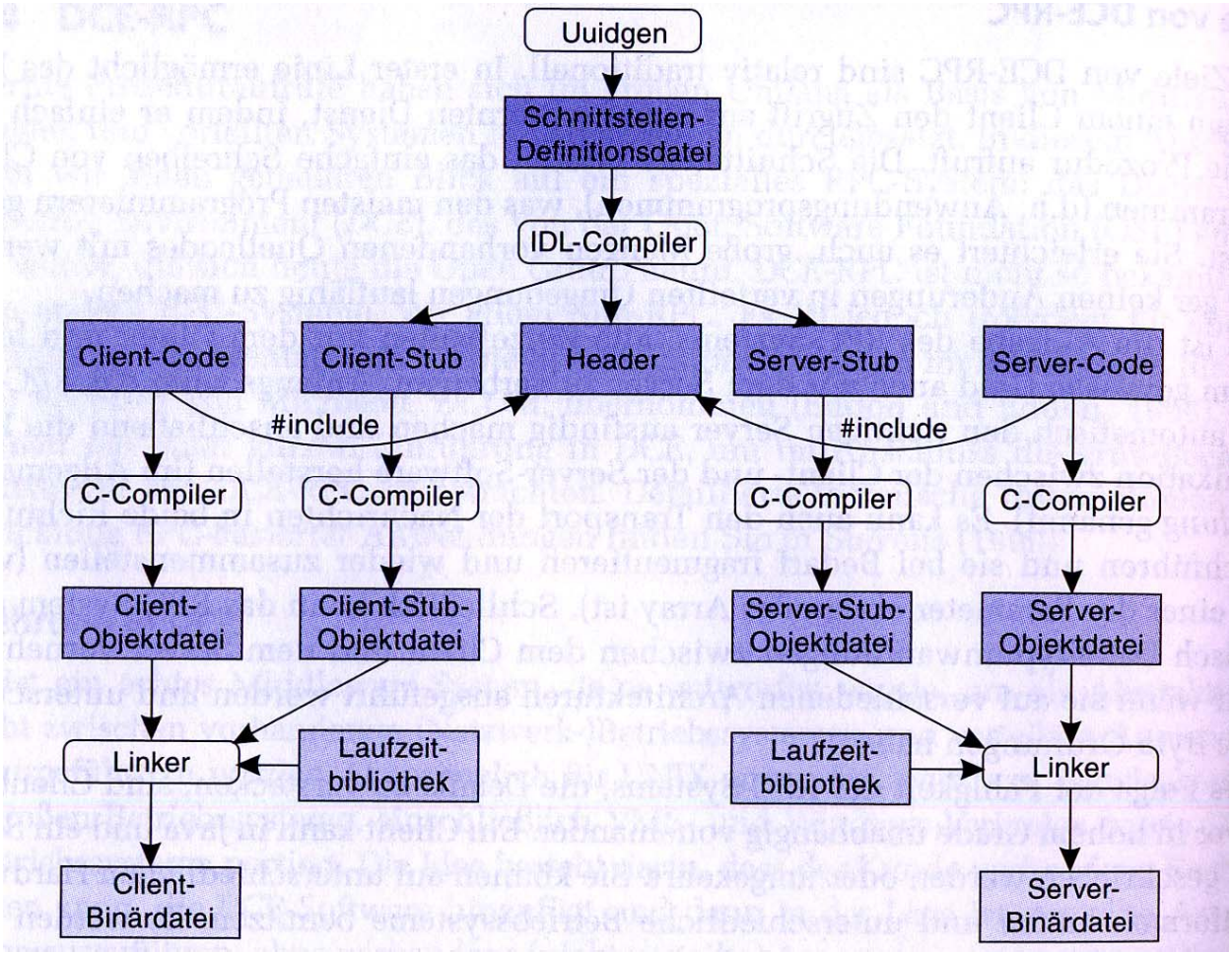
Dies kann unter Umständen bei Systemen Probleme verursachen, in denen unterschiedliche Maschinen zum Einsatz kommen -> zB verschiedene Zeichensätze (ASCII, Little Indian, Big Indian).



12)
Wie schreibt man für RPCs Client und Server und welche Rolle spielt dabei die IDL? Welche Ziele werden mit dem Einsatz einer IDL in einem verteilten System verfolgt? Was versteht man in diesem Zusammenhang unter "binding"?

Wie schreibt man für RPCs Client und Server und welche Rolle spielt dabei die IDL?

Schritte beim Schreiben eines Clients und eines Servers:



Die IDL-Datei ist ein global eindeutiger Bezeichner für die angegebene Schnittstelle. Mit dem IDL-Compiler wird Sie als Schnittstellendefinition in die jeweilige Programmiersprache übersetzt. Die IDL erzeugt den Client-Stub und den Server-Stub. Stellt auch sicher, dass der Client mit dem richtigen Server verbunden ist, da bei abweichender Schnittstelle eine NICHT-Übereinstimmung erkannt wird.

Welche Ziele werden mit dem Einsatz einer IDL in einem verteilten System verfolgt?

Die Verwendung von IDL vereinfacht deutlich die Realisierung von RPC, da mit dem dazugehörigen Compiler einfach der Server-Stub und der Client-Stub erstellt wird. Der IDL-Compiler erzeugt folgende Dateien:

- Header-Datei (wird jeweils in den Server- und Client-code inkludiert)
- Client-Stub
- Server-Stub

Was versteht man in diesem Zusammenhang unter "binding"?

Damit ein Client einen Server aufrufen kann, muss der Server registriert und zur Annahme eingehender Anforderungen vorbereitet werden. Das Registrieren des Servers ermöglicht es einem Client, ihn aufzufinden und sich an ihn zu binden. Das Auffinden des Servers erfolgt in zwei Schritten:

- Auffinden des Server-Computers
- Auffinden des Servers (dh. des richtigen Prozesses) auf dem Computer

Konkret benötigt der Client einen Server-Endpunkt, mit dem er kommunizieren kann. Betriebssysteme bezeichnen solche Punkte Ports um Nachrichten für verschiedene Prozesse auseinanderzuhalten.

13)
Welche Arten von asynchronen RPCs gibt es? Geben Sie auch einen verallgemeinerten Überblick über verschiedene Typen der Kommunikation (persistent/transient bzw. synchron/asynchron).

Welche Arten von asynchronen RPCs gibt es?

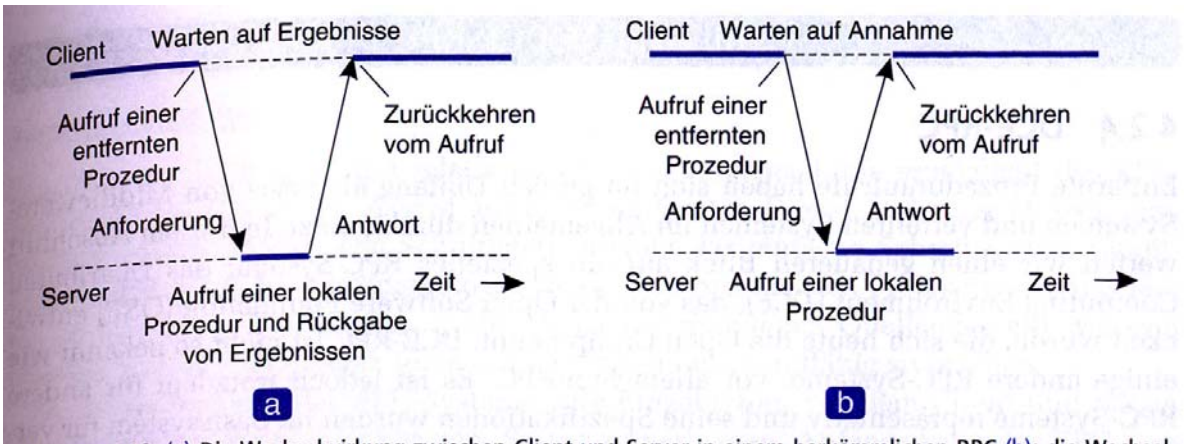


Abbildung 4.10: (a) Die Wechselwirkung zwischen Client und Server in einem herkömmlichen RPC (b); die Wechselwirkung bei asynchronem RPC

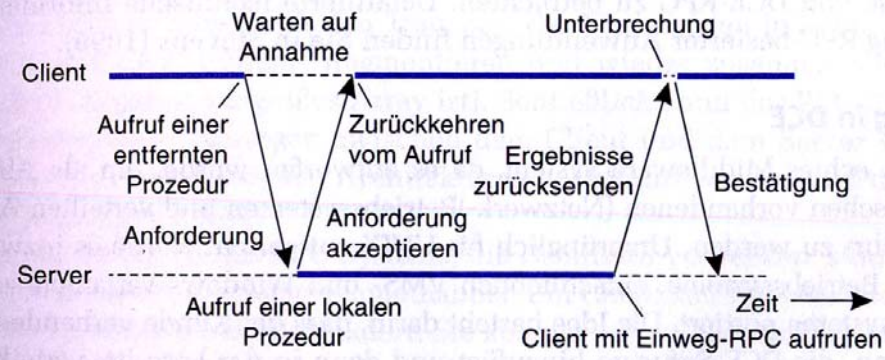
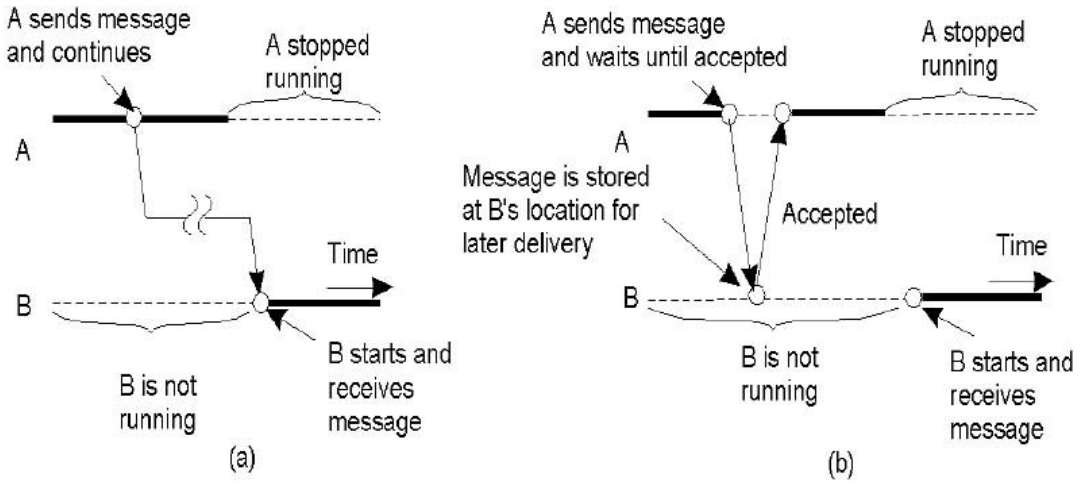


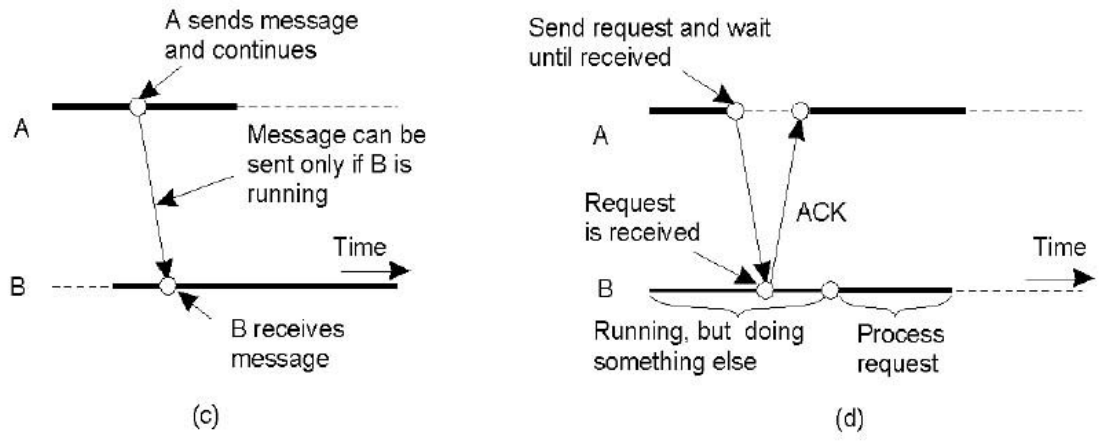
Abbildung 4.11: Ein Client und ein Server kommunizieren über zwei asynchrone RPCs.

- **Bei asynchronen RPCs** entsteht durch den entfernen Aufruf eine zeitliche Verzögerung. Erst wenn der Server an den Client ein „ACCEPT“ zurückschickt, wird mit der Verarbeitung fortgesetzt.
- **EINWEG RPC**
  -> Client setzt unmittelbar nach absenden der Anforderung fort, und wartet NICHT auf eine Bestätigung des Servers
- **VERZÖGERTER SYNCHRONER RPC (siehe 4.11)**
  -> ist eine Zusammenfassung zweier asynchroner RPCs. Dabei kann zwischen Antwort ACCEPT eine weitere Verarbeitung stattfinden, die unabhängig von der Serverantwort ist.

<u>Geben Sie auch einen verallgemeinerten Überblick über verschiedene Typen der Kommunikation (persistent/transient bzw. synchron/asynchron).</u>



a) Persistent asynchronous communication
b) Persistent synchronous communication



c) Transient asynchronous communication
d) Receipt-based transient synchronous communication

PERSISTENTE KOMMUNIKATION
besagt, dass die Nachricht im Kommunikationssystem gespeichert ist.

TRANSIENTE KOMMUNIKATION
besagt, dass die Nachricht nur gespeichert wird, wie die sendende und empfangende Anwendung ausgeführt wird.

SYNCHRONE KOMMUNIKATION

Der Sender ist blockiert bis:

- Bis Nachricht an Empfänger geliefert
- Empfänger Nachricht verarbeitet hat
- Buffer am Empfängerhost ist frei

Der Sender ist gesperrt, bis er weiß, dass seine Anforderung akzeptiert wurde.

ASYNCHRONE KOMMUNIKATION

Der Sender fährt unmittelbar nach der Übertragung fort.

14)
Erläutern Sie die Grundprinzipien verteilter Objekte sowie der Remote Object (bzw. Method) Invocation. Gehen Sie auf die Begriffe "proxy" und "skeleton" ein. Erklären Sie den Unterschied zwischen "Compile-time" und "Run-time" Objekten. Erklären Sie den Unterschied zwischen persistenten und transienten Objekten.

Erläutern Sie die Grundprinzipien verteilter Objekte sowie der Remote Object (bzw. Method) Invocation.

Ein verteiltes Objekt realisiert als Grundlage die Trennung zwischen Schnittstelle und Objektimplementierung. Diese Trennung ermöglicht es uns, die Schnittstelle auf einen Rechner zu legen und das Objekt selbst auf einen anderen. Dieser Aufbau wird als verteiltes Objekt bezeichnet.

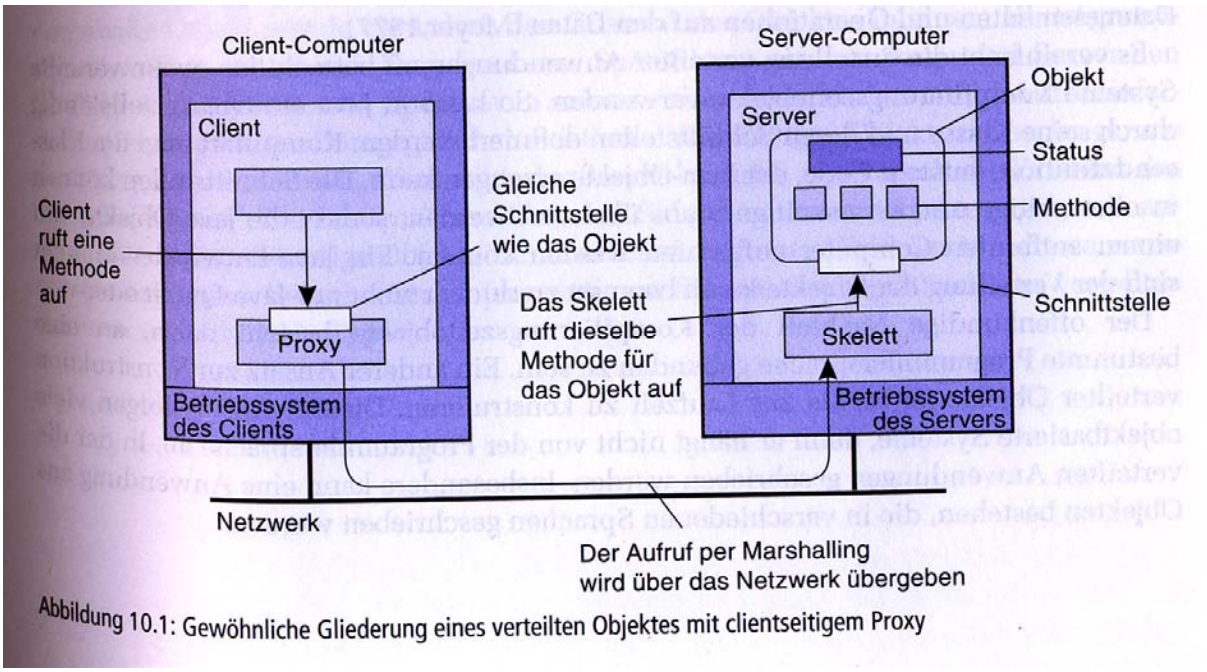


Abbildung 10.1: Gewöhnliche Gliederung eines verteilten Objektes mit clientseitigem Proxy

<u>RMI:</u>

RMI und RPC unterscheiden sich vor allem darin, dass RMI systemweite Objektreferenzen anbietet (Objektserver). Der Client bindet sich an ein entferntes Objekt, dabei wird die Proxy (Client-Stub) geladen.

Gehen Sie auf die Begriffe "proxy" und "skeleton" ein

PROXY ist der Client-Stub
Diese übernimmt das Marshalling der Parameter sowie das Unmarshalling der Antwort.

SKELETON ist der Server-Stub
Übernimmt das Parameterunmarshalling, das Ausführen des Methodenaufrufs des Objekts und Marshalling der Antwort am Server.

Erklären Sie den Unterschied zwischen "Compile-time" und "Run-time" Objekten.

Compile-Time Objekte sind Objekte die als Instanzen von Klassen geführt werden.

Run-Time-Objekte sind unabhängig von einer bestimmten Implementierung/Programmiersprache. Sie werden zur Laufzeit erzeugt und mit Hilfe eines Objektadapters an den Client angepasst. Die Objekte werden nur aufgrund ihrer Schnittstelle definiert.

Erklären Sie den Unterschied zwischen persistenten und transienten Objekten.

PERSISTENTE OBJEKTE

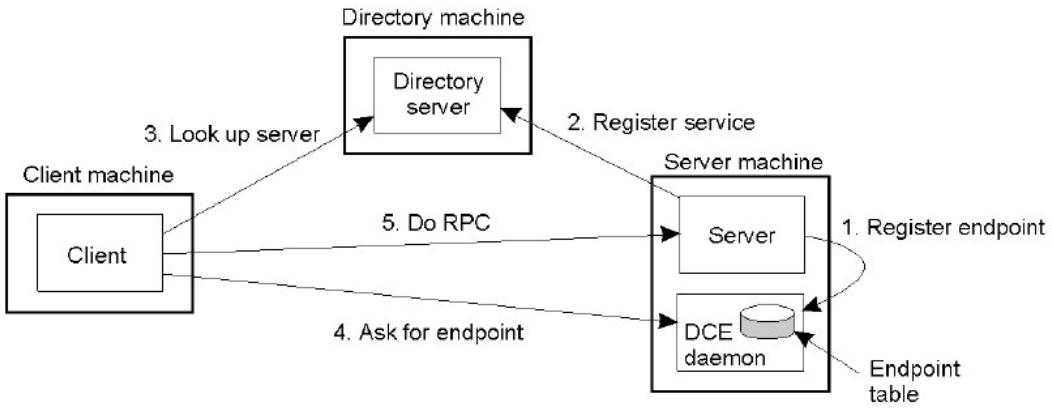
Werden dauerhaft gepeichert, unabhängig vom den Serverprozessen. Nach Neustart können diese wiederhergestellt werden.

TRANSIENTE OBJEKTE

Existieren nur im Arbeitsspeicher des Servers, dh wenn der Server beendet wird hört das Objekt auf zu existieren.

15)
Wie funktioniert das Binding bei RMI? Welchen Zusammenhang gibt es zu den verschiedenen Arten, eine object reference zu implementieren. Vergleichen Sie (exemplarisch) CORBA und Java in Bezug auf Objektreferenzen.

Wie funktioniert das Binding bei RMI?



1. Server registriert einen Endpoint
2. Server registriert sich beim Directory Server
3. Client looks up the Service at a Directory Service
4. Client asks Server for Endpoint
5. Client sends RPC

Binding führt dazu, dass der Client-Stub (Proxy) angelegt wird, der eine Schnittstelle mit Methoden zur Verfügung stellt, der die Methoden des entfernten Objekts aufrufen.

Welchen Zusammenhang gibt es zu den verschiedenen Arten, eine object reference zu implementieren.

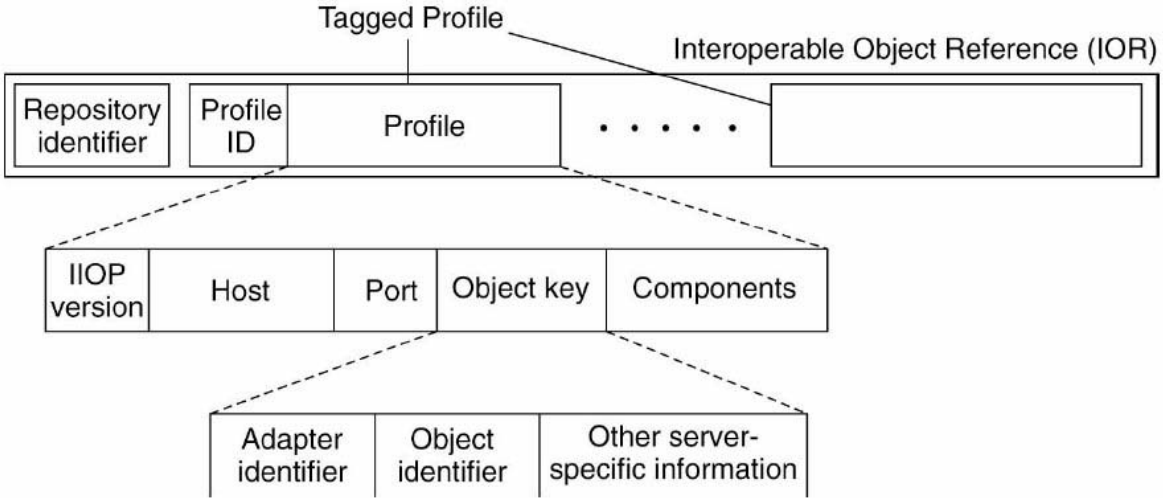
Das Objekt muss in jedemfall eindeutig innerhalb einer Domäne identifizierbar sein.

- Network address + endpoint (+ object)
- Network address + server ID (daemon)
- Location server + unique server ID
- …

Vergleichen Sie (exemplarisch) CORBA und Java in Bezug auf Objektreferenzen.

CORBA

In Corba wird die IOR standardisiert verwendet, um Objekte in Systemen zu registrieren. Um diese zu erhalten, kann der Client einen Naming Service verwenden.



The organization of an IOR with specific information for IIOP.

JAVA

Bei Java kann man zwei Typen von Binding unterscheiden:

- IMPLIZIT
  - ist transparent, Bindung durch auflösen der Referenz (Zeiger) auf das Objekt
- EXPLIZIT
  - dabei muss zuerst eine bestimmte Funktion aufgerufen werden, die einen Zeiger liefert, ehe man dann diese Referenz zum Objekt auflösen kann.

16)
Was versteht man unter "static" und "dynamic" Invocation von verteilten Methoden? Geben Sie Beispiele an.

STATIC INVOCATON
setzt voraus, dass die Schnittstellen/Methoden eines Objektes bereits bekannt sind. Eine Änderung erfordert eine Neukompilierung des Clients.

Bsp: MyObject.doSomething(in1,in2,out1, out2)

DYNAMIC INVOCATION
erlaubt es den Namen der Mehode des verteilten Objekes, die es aufrufen will, zur Laufzeit zu bestimmen.

Bsp: invoke(MyObject, id(doSomething), in1, in2, out1, out2)
id() bezeichnet hier den Look-up nach dem Methodennamen (zB. Aus einem Directory Service)

Beispielsweise kann eine Software über einen Objektbrowser Objekte auf entfernten Systemen untersuchen und dann dynamisch eine Methode (über Auswahl) aufrufen.

Anwendungen von dynamic invocation:

- ObjektBrowser
- Stapelverarbeitungsdienst
- Batch-Verarbeitung
- Frameworks

17)
Wie funktioniert die Parameterübergabe bei RMI? Gehen Sie dabei auf jene Eigenschaften der Objektorientierung ein, welche den Vorteil von RMI gegenüber RPC bewirken.

Wie funktioniert die Parameterübergabe bei RMI?

Es wird bei der Übergabe zwischen lokalen Objekten und entfernten verteilten Objekten getroffen.

Beim Aufrufen einer Methode, die als Parameter eine Objektreferenz übergibt wird dieser Parameter nur dann als Referenz übergeben, wenn es sich um ein entferntes Objekt handelt. Stellt sich heraus, dass die Objektreferenz auf ein lokales Objekt zeigt, wird das lokale Objekt kopiert und als Ganzes übergeben.

In Java-RMI können nur serialisierbare Objekte vom Proxy an das Skeletton übermittelt werden.

Gehen Sie dabei auf jene Eigenschaften der Objektorientierung ein, welche den Vorteil von RMI gegenüber RPC bewirken.

Bei RPC können keine Objektreferenzen als Parameter übergeben werden. Auch ist es nicht möglich Zeiger (Programmiersprache C) zu übergeben. Es werden diese Parameter mittels Copy/Restore übergeben.

In RMI können auch Objektreferenzen übergeben werden, wodurch über eine Registry wiederum ein weiteres entferntes Objekt aufgerufen werden kann.

18)
Erläutern Sie die Grundprinzipien von Message-orientierter Kommunikation und gehen Sie auf CORBA Messaging exemplarisch ein. Beschreiben Sie zwei unterschiedliche Methoden, wie asynchrone Methodenaufrufe in CORBA Messaging erfolgen können.

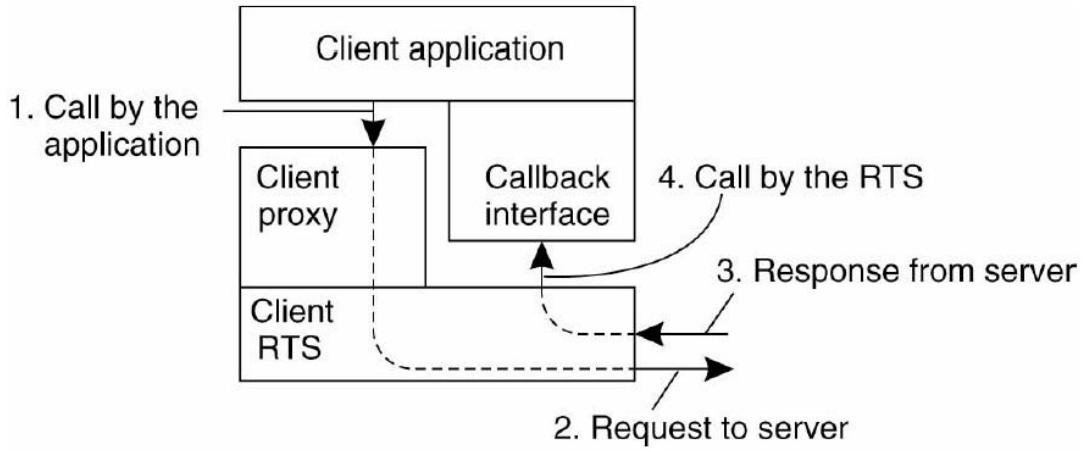
Erläutern Sie die Grundprinzipien von Message-orientierter Kommunikation und gehen Sie auf CORBA Messaging exemplarisch ein.

- RPC und Verteilte Objekte erweitern die gewohnte Programmierung an die verteilte Umgebung und machen entfernte Kommunikation (implizit) transparent.
- Nachrichtenorientierte Middleware (MOM) führt ein anderes Modell ein, für eine verteilte Programmierung – EXPLIZITE Aufrufe (Kommunikation)
- Kommunikationsmechanismen können sein synchron, asyncrhon, persistent und transient.

CORBA unterscheidet sich von anderen Systemen dadurch, dass sie die Nachrichtenübermittlung konsequent **objektbasiert** handhabt. Sämtliche Kommunikation erfolgt durch den Aufruf von Objekten.

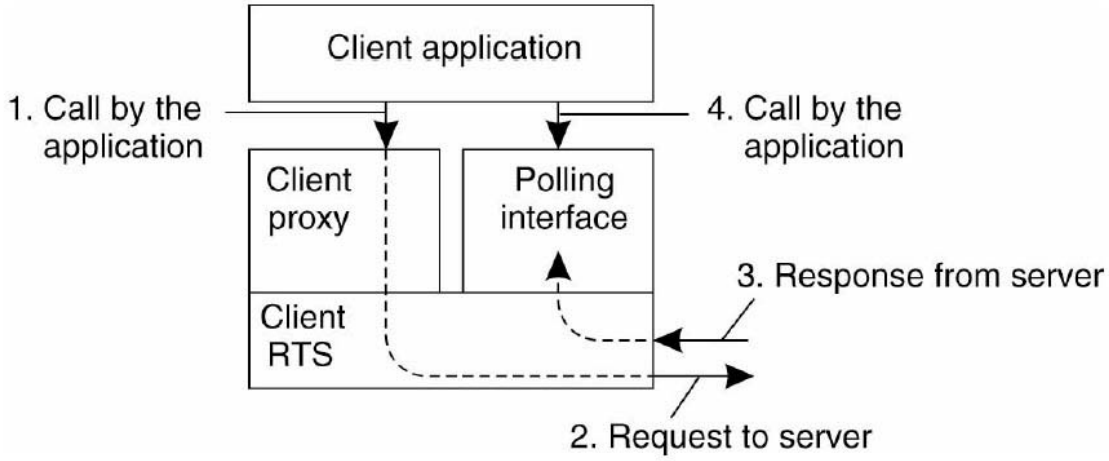
Beschreiben Sie zwei unterschiedliche Methoden, wie asynchrone Methodenaufrufe in CORBA Messaging erfolgen können.

- CALL-BACK Modell
  Der Aufrufer (Client) implementiert eine Reihe Call-Back Methoden, an die die Antwort/Ergebnis übergeben wird -> verzögert, asynchron



CORBA's callback model for asynchronous method invocation.

- POLLING Modell
  Es werden zur ursprünglichen Schnittstelle 2 neue Schnittstellen erzeugt. Eine dient dem Aufruf und die andere ruft das Polling Interface ab und prüft auf eingegangene Antworten. Das ist muss von der Antwendung explizit angestossen werden. Dadurch wird eine asynchrone Kommunikation mit dem Server erreicht.



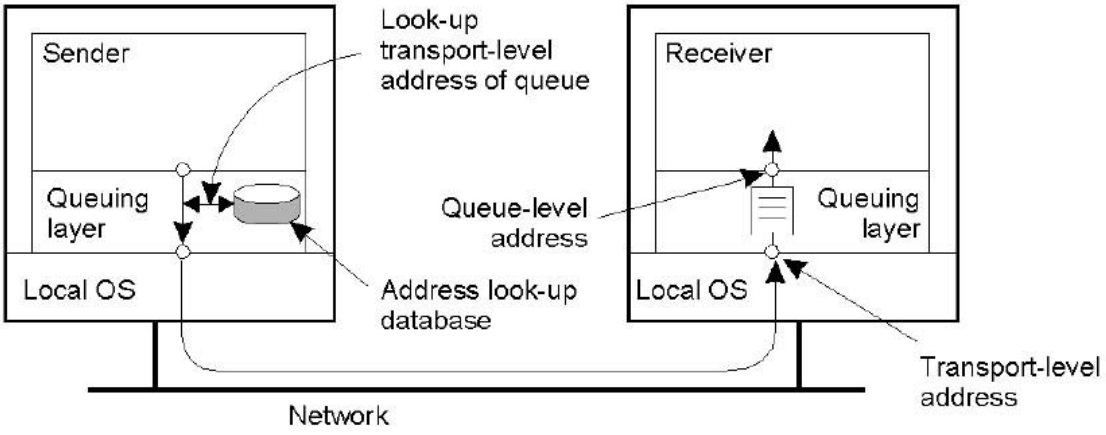
CORBA's polling model for asynchronous method invocation.

19)
Was versteht man unter "Message-oriented Middleware MoM"? Erläutern Sie Modell und Architektur solcher "Message-Queueing"-Systeme. Erklären Sie die Primitivoperationen Put, Get, Poll und Notify eines Message-Queuing Systems. Diskutieren Sie Einsatzzwecke sowie Vor- und Nachteile - gehen Sie insbesonders auf den Begriff des Message Brokers und dessen Bedeutung für EAI ein.

Was versteht man unter "Message-oriented Middleware MoM"?

- Message oriented Middleware bietet Unterstützung für persistente asynchrone Kommunikation.
- Anwendungen kommunizieren, indem sie ihre Nachrichten in eine Message Queue einstellen.
- Es gibt nur die Garantie für den Sender, dass eine Nachricht in eine Queue eingestellt wurde.

Erläutern Sie Modell und Architektur solcher "Message-Queueing"-Systeme.



The relationship between queue-level addressing and network-level addressing.

Erklären Sie die Primitivoperationen Put, Get, Poll und Notify eines Message-Queuing Systems.

- GET
  blockt solange die Queue leer ist, gibt ansonsten die erste Nachricht zurück.
- PUT
  hängt die Nachricht an eine Queue an
- POLL
  Queue nach Nachrichten durchsuchen, gibt die erste Nachricht zurück. Blockt niemals.
- NOTIFY
  installiert einen Handler, der aufgerufen wird, wenn eine Nachricht in die Queue gestellt wird.

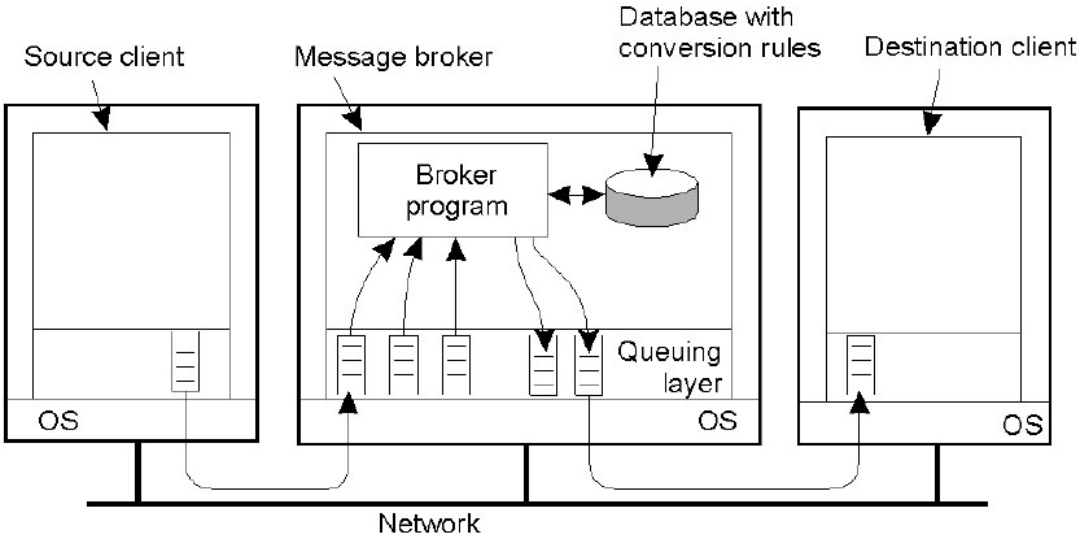
Diskutieren Sie Einsatzzwecke sowie Vor- und Nachteile - gehen Sie insbesonders auf den Begriff des Message Brokers und dessen Bedeutung für EAI ein.

- Email
- Workflow
- Groupware
- Stapelverarbeitung

Ein Message Broker wird verwendet, um Nachrichten von einem Format in ein anderes zu konvertieren (Database mit Konvertierungsregeln) -> Konvertierungstool.

EAI – Enterprise Application Integration -> Anwendungen unabhängig von ihrer Datenbank.

RPC und RMI haben den Nachteil, das Aufrufer und Aufgerufener zum Kommunikationszeitpunkt beide aktiv sein müssen. Diese enge Kopplung hat sich häufig als schwerwiegender Nachteil herausgestellt und zu Warteschlangensystemen geführt oder MOM.



20)

Erklären Sie "stream-oriented communication". Was ist "QoS" und inwiefern ist es für stream-oriented communication von Bedeutung?

Erklären Sie "stream-oriented communication"

- Ein Datenstream ist nichts weiter als eine Folge von Dateneinheiten.
- Sie können sowohl auf kontinuierliche (Audio, Video), als auch auf diskrete Medien angewendet werden.
- Die zeitliche Abfolge ist für das Streaming von größter Bedeutung, damit die Daten rechtzeitig beim Client eintreffen.

- asynchron: keine zeitliche Beschränkung (Dateidownload)
- Synchron: maximale End-to-End Verzögerung (Sensoren)
- Isochron: minimale und maximale End-To-End Verzögerung (Audio/Video Stream)

Ein einfacher Stream besteht aus einer Datenfolge (zB Mono Audio)

Ein komplexer Stream beinhaltet mehrere Substreams die in zeitlicher Abhängigkeit zueinander stehen (Stereo-Audio, Video mit Tonspuren)

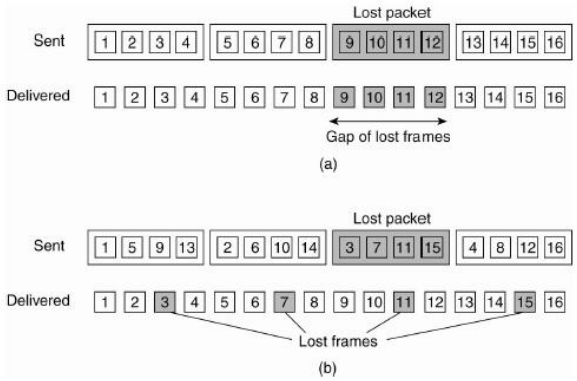
Was ist "QoS" und inwiefern ist es für stream-oriented communication von Bedeutung?

Qualitätskriterien:

- Erforderliche BIT-RATE
- Max. Verzögerung bei Session beginn
- Max. End-to-End Verzögerung
- Max. Jitter
- Max. Umlaufverzögerung

Um Jitter zu vermeiden, wird ein Buffer verwendet. Somit können kleine Unregelmäßigkeiten im Datenstrom ausgeglichen werden.

Wenn ein Packet mehrere (Audio und/oder Video) Frames enthält und verloren geht, dann kann eine große Lücke beim Abspielen entstehen. Durch verschachtelte Rahmen (interleaved transmission) kann dieser Effekt reduziert werden. Dafür muss aber eine größere Startzeit in Kauf genommen werden.



21)
Was ist der Unterschied zwischen Prozess und Thread? Was ist beim Einsatz von Multi-Threading zu beachten? Welche Bedeutung haben Threads in verteilten Systemen, insbesonders in Client/Server-Umgebungen?

Was ist der Unterschied zwischen Prozess und Thread?

PROZESS

- Ein Prozess ist ein Programm in Ausführung und wird vom Betriebssystem auf einem eigenen virtuellen Prozessor ausgeführt
- Er hat einen eigenen Adressraum und kann so nicht auf den Speicher anderer Prozesse zugreifen (geschützt)
- Das Erstellen eines Prozesses ist relativ aufwändig. Es muss ein eigener Adressraum generiert werden und dieser gereinigt werden (Alle Bits auf 0 setzen)

THREAD

- Ein Prozess kann aus mehreren Threads bestehen
- Threads teilen sich den selben Adressraum wie der Prozess
  -> hierbei muss der Programmierer dafür sorgen, dass sich konkurrierende Threads nicht gegenseitig in den selben Speicher schreiben und so unerwünschte Zustände erzeugen
- Beim Auftreten eines System-Calls blockiert nur der betroffene Thread, und nicht der Prozess in dem er lauft selbst.

UNTERSCHIEDE ZWISCHEN THREADS UND PROZESSEN

- Das switchen zwischen Prozessen ist teurer als das switschen unter Threads
  -> bei Prozessen muss der CPU-Kontext sowie Adressraum, offene Dateien etc. gewechselt werden
  -> bei Threads lediglich der CPU-Kontext
- Blockiert ein Thread, so kann einfach in den nächsten gewechselt werden -> günstiger zu Programmieren als mit Prozessen

Was ist beim Einsatz von Multi-Threading zu beachten?

- Threads sollen terminieren
- Threads müssen in Bezug auf Speicher synchronisiert werden (Locks)
- Effizienter und sorgsamer Umgang mit Threads

Welche Bedeutung haben Threads in verteilten Systemen, insbesonders in Client/Server-Umgebungen?

- Ermöglicht Verteilungstransparenz (verbergen von Latenzzeiten/Wartezeiten auf Antwort)
- Höherer Durchsatz
- Aufteilen der Last, DIVIDE&CONQUER – Jeder bearbeitet nur einen Teil des Gesamten.

22)

Welche Aspekte Verteilter Systeme sind auf Client-Seite zu berücksichtigen? Wie werden User Interfaces in die Architektur Verteilter Systeme eingebunden? Wie können dabei verschiedene Arten der Transparenz unterstützt werden?

Welche Aspekte Verteilter Systeme sind auf Client-Seite zu berücksichtigen?

- Parallele Interaktion zwischen Mensch auf Client-Seite und Server (Kommunikation blockiert nicht mit Server wegen Usereingabe)
- Kommunikationslatenzzeiten verbergen (Multi-Threading)
- Parallele Verarbeitung am Server (Verbindungen zu verschiedenen Replikas, parallele Datenübertragung)
- Lokale Datenverarbeitung (Thik Client)
- Komponenten zur Unterstützung von Transparenz

Wie werden User Interfaces in die Architektur Verteilter Systeme eingebunden?

- Mehrschichtenarchitektur (N-Tier)
  -> Trennung von User-Interface, Applikationslogik und Datenhaltung

Wie können dabei verschiedene Arten der Transparenz unterstützt werden?

Neben der Benutzeroberfläche und anderer für die Applikation benötigter Software besteht die Client-Software aus Komponenten, mit denen Verteilungstransparenz erzielt werden soll.

Im Idealfall soll ein Client nicht erkennen, dass er mit einem entfernten Objekt kommuniziert.

- **ZUGRIFFSTRANSPARENZ**
  wird häufig realisiert, indem aus einer Schnittstellendefinition ein Server-Stub und ein Client-Stub erzeugt wird. Entfernte Aufrufe werden wie lokale behandelt.
- **NEBENLÄUFIGKEITSTRANSPARENZ**
  -> Transaktionsmonitor
- **ORTS-, MIGRATIONS-, und RELOKATIONSTRANSPARENZ**
  -> Verwendung von praktischen Namenssystemen
  -> Bei Bindung eines Clients an einen Server kann der Client vom Server bei einer Positionsänderung informiert werden
  -> Proxy (Relokatonstransparenz)
- **FEHLERTRANSPARENZ**
  -> Die Client-Middleware kann so konfiguriert werden, dass sie wiederholt versucht zu einem Server eine Verbindung aufzubauen bzw. andere Server auszuprobieren (zB Emule)

23)
Geben Sie grundlegende Design-Entscheidungen für Server an und bewerten Sie diese. Gehen Sie auf den Unterschied zwischen stateful und stateless Servern genauer ein und geben Sie Beispiele an. Erläutern Sie anhand einer Skizze Architektur und Funktionsweise eines multi-threaded Servers (zB File- oder Web-Server). Was ist beim Einsatz von Multi-Threading vom Entwickler besonders zu beachten?

Geben Sie grundlegende Design-Entscheidungen für Server an und bewerten Sie diese.

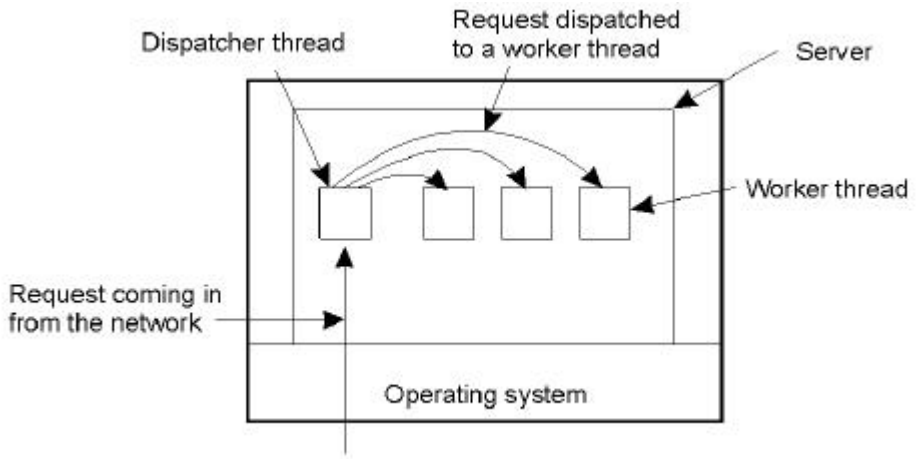
- Interative Programmierung oder Parallele (Concurrent/MultiThreading)
- Bei Übertragungsunterbrechung vom Client zum Server den Client über OUT-OF-BAND Kontrolle steuern (Zusätzlicher Steuerendpunkt zwischen Client/Server)
- Zustandslose oder Zustandsbehaftete Server (Halten oder nicht Clientinformationen)
- Serversuche/Binding (Ports, Namensservice, Verzeichnisdienst)
- Multithreaded Servers
  -> Parallele Verarbeitung ausnutzen

Gehen Sie auf den Unterschied zwischen stateful und stateless Servern genauer ein und geben Sie Beispiele an.

STATEFUL SERVER
Hält Informationen über seine Clients (zB Dateiserver, Client Zugriffsberechtigungen)

STATELESS SERVER
Hält keine Informationen über seine Clients (zB Webserver, bedient lediglich Anfragen)

Erläutern Sie anhand einer Skizze Architektur und Funktionsweise eines multi-threaded Servers (zB File- oder Web-Server).



Was ist beim Einsatz von Multi-Threading vom Entwickler besonders zu beachten?

Multithreaded Servers bestehen aus einem dispatcher und mehreren worker threads.

Der DISPATCHER Thread wartet auf eingehende Anfragen, und startet pro Anfrage einen WORKER Thread, an welchen die Anfrage weitergereicht wird. Damit ist es möglich, dass während eine Anfrage bearbeitet wird, der Dispatcher Thread auf eine neue reagiert.

24)
Was sind die Besonderheiten von Objekt-Servern? Welche Arten gibt es dabei für die "Invocation", also den Aufruf (evtl. auch die Aktivierung/Activation) eines Objektes auf Server-Seite (Policies hinsichtlich thread, code sharing, und object creation)? Was ist in diesem Zusammenhang ein Objekt-Adapter?

Was sind die Besonderheiten von Objekt-Servern?

- Ein Objektserver dient dem Bereitstellen von Objekten in einem verteilten System. Im Gegensatz zu herkömmlichen Servern, stellt der Objektserver keine spezifischen Dienste zur Verfügung. Dafür sind die Objekte des Servers zuständig.
- Er stellt lediglich Mittel zur Verfügung, um seine Objekte „remote“ zugreifbar zu machen.
- Um ein Objekt über den Server zugreifbar zu machen, muss es zuerst in seinen Adressraum gebracht werden -> siehe verschiedene Arten der Aktivierung

Welche Arten gibt es dabei für die "Invocation", also den Aufruf (evtl. auch die Aktivierung/Activation) eines Objektes auf Server-Seite (Policies hinsichtlich thread, code sharing, und object creation)?

AKTIVIERUNGSRICHTLINIEN

- THREAD
  wird ein eigener Thread beim Aufruf gestartet oder nicht
- CODE SHARING
  Objekte können sich den Code teilen, sodass sie nur einmal am Server geladen werden müssen, Objekte teilen sich einen Speicherbereich
- OBJECT CREATION
  persistente (permanente) oder transiente (flüchtige) Objekte

Was ist in diesem Zusammenhang ein Objekt-Adapter?

- Ein Objekt-Adapter dienst als Hüller um ein Objekt und stellt einen Mechanismus zur Verfügung um ein Objekt zu AKTIVIEREN
- Ein Objekt-Adapter ist GENERISCH, dh er weiß nichts von der Schnittstelle seiner Objekte und kann so für beliebige Objekte verwendet werden. Er kann mehrere Objekte beinhalten.

Erläutern Sie die wichtigsten Aspekte der Code Migration. Erklären Sie "strong mobility" und "weak mobility" und geben Sie für "weak mobility" ein Beispiel an.

Erläutern Sie die wichtigsten Aspekte der Code Migration.

- Übergabe von Programmen (Programmteilen)
- Übergabe während der Ausführung
- Umgang mit lokalen Ressourcen bei Ausführung eines migrierten Programms
- Ist im weitesten Sinne die Verschiebung von Programmen von einem Computer auf den anderen

CODESEGMENT
ist der Teil, der den Befehlssatz enthält, aus dem sich das auszuführende Programm zusammensetzt.

RESSOURCENSEGMENT
enthält Referenzen auf externe Ressourcen, die vom Prozess gebraucht werden.

AUSFÜHRUNGSSEGMENT
wird verwendet um den aktuellen Ausführungszustand eines Prozesses zu speichern (private Daten, Stack, Programmzähler)

Erklären Sie "strong mobility" und "weak mobility" und geben Sie für "weak mobility" ein Beispiel an.

STRONG MOBILITY

Dabei wird auch das Ausführungssegment übertragen. Dh der Prozess wird in seinem aktuellen Zustand verschoben.

WEAK MOBILITY

Es wird nur das Codesegment übergeben, vielleicht einige Initialisierungdaten.

Beispiel: Java-Applets

25)
Erläutern Sie das Konzept der Virtualisierung und in weiterer Folge deren Bedeutung für die Code Migration in heterogenen Umgebungen. Beschreiben Sie die zwei verschiedenen Arten von Architekturen von "virtual machines".

Erläutern Sie das Konzept der Virtualisierung und in weiterer Folge deren Bedeutung für die Code Migration in heterogenen Umgebungen.

Bei Virtualisierung geht es darum, eine bestehende Schnittstelle so zu erweitern oder zu ersetzen, dass sie das Verhalten eines anderen Systems nachahmt.

Die Migration bietet eine Form der starken Mobilität bei der Codemigration. Beispielsweise werden Java-Applets auf einer JVM (Java Virtual Machine) auf beliebigen Clients ausgeführt. Die JVM bietet eine Plattformübergreifende Schnittstelle für Java-Code.

Bei Virtualisierung wird di

Beschreiben Sie die zwei verschiedenen Arten von Architekturen von "virtual machines".

VIRTUELLE PROZESSMASCHINE
Erstellung eines Laufzeitsystems, dass einen abstrakten Befehlssatz zum Ausführen von Anwendungen bereitstellt. Diese Befehlt werden interpretiert oder emuliert.

VIRTUAL MACHINE MONITOR
Stellt eine Schicht bereit, die die ursprüngliche Hardware vollständig überdeckt bzw. abschirmt. zB VMWare

26)
Erläutern Sie die Begriffe "Name", "Identifier", "Address" sowie den Bezug zwischen diesen Begriffen in der Praxis.

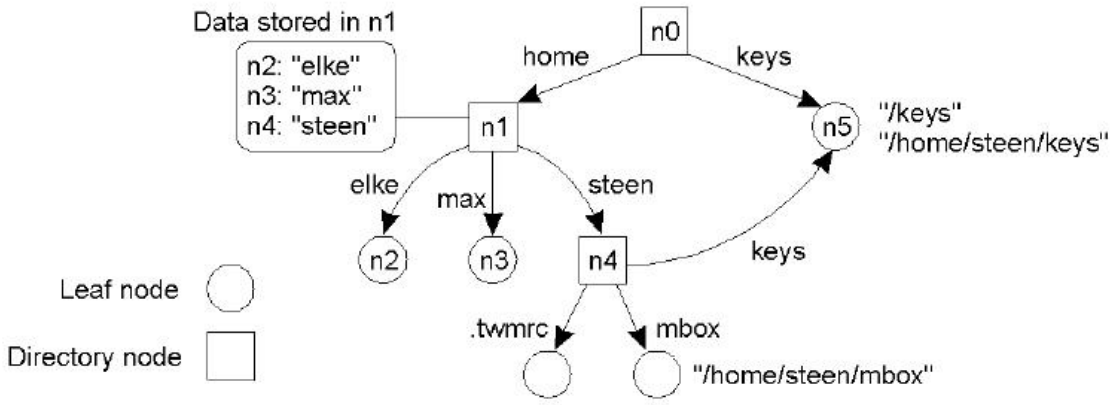
Diese Begriffe stellen ZUGRIFFSPUNKTE dar.

- **NAME**
  benennt eine Entität -> meist Benutzerfreundliche (unabhängig von Adresse durch Namensauflösung)
- **ADDRESS**
  Ist der Name/Ort eines Zugriffspunktes einer Entität.
- **IDENTIFIER**
  für eindeutige Kennzeichnung einer Entität, die folgenden Kriterien entspricht:
  -> Jeder Identifier verweist höchstens auf eine Entität
  -> Auf jede Entität verweist höchstens ein Identifier
  -> Ein Identifier verweist immer auf die gleiche Entität

27)
Was ist ein "Name Space"? Erläutern Sie das Grundprinzip des "Closure Mechanismus" anhand eines Beispieles (zB Unix File System).

Was ist ein "Name Space"?

- Strukturiert Namen
- Hierachische Namen können als **beschriftete gerichtete** Graphen dargestellt werden
  -> BLATTKNOTEN (=benannte Entität)
  -> VERZEICHNISKNOTEN (hat ausgehende Kanten, speichert Verzeichnistabelle)



Erläutern Sie das Grundprinzip des "Closure Mechanismus" anhand eines Beispieles (zB Unix File System).

CLOSURE MECHANISMUS

Ist nichts anderes als der **Startpunkt** in einem Namensbaum. Identifiziert das Root-Verzeichnis.

Um absoluten Verzeichnispfad (/users/home) aufzulösen, muss dem Dateisystem der Root-Knoten „/“ bekannt sein. Diese Information ist im Superblock des logischen Laufwerks gespeichert.

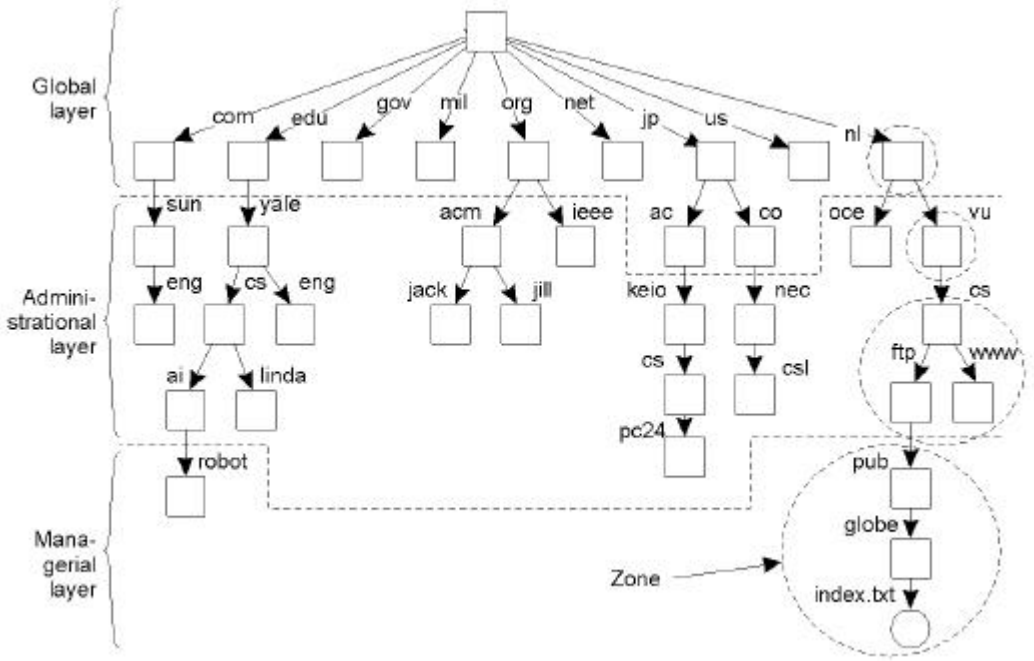
28)
Erklären Sie die Schichten der Verteilung von Name Spaces. Erläutern Sie die Einsatzmöglichkeiten von Replication und Caching in den verschiedenen Schichten. Erklären Sie verschiedene (hierarchische) Möglichkeiten der iterativen/rekursiven "name resolution".

Erklären Sie die Schichten der Verteilung von Name Spaces.

- GLOBALE SCHICHT
  -> wird von Knoten höchster Ebene gebildet
  -> .com, .edu etc. im DNS
- ADMINISTRATIONSSCHICHT
  -> wird von Verzeichnisknoten gebildet, die innerhalb einer Organisation verwaltet werden
  -> zB www.xxx.vu.nl im DNS
- MANAGEMENTSCHICHT
  -> besteht aus Knoten, die regelmäßig Änderungen unterliegen. Zu dieser Schicht gehören beispielsweise Knoten, die Hosts im lokalen Netzwerk darstellen.
  -> zB /pub/globe/index.txt in einer ftp-URL.

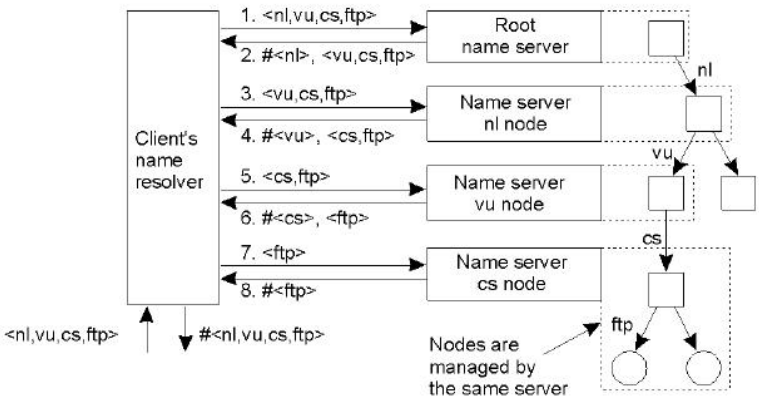
Erläutern Sie die Einsatzmöglichkeiten von Replication und Caching in den verschiedenen Schichten.

- GLOBALE SCHICHT
  - Effektives Caching durch Clients
  - Viele Replikate (da wenig Änderungen)
- ADMINISTRATIONSSCHICHT
  - Caching durch Clients möglich
  - keine bis wenige Replikate
- MANAGEMENT SCHICHT
  - keine Replikate
  - Caching durch Clients manchmal möglich#

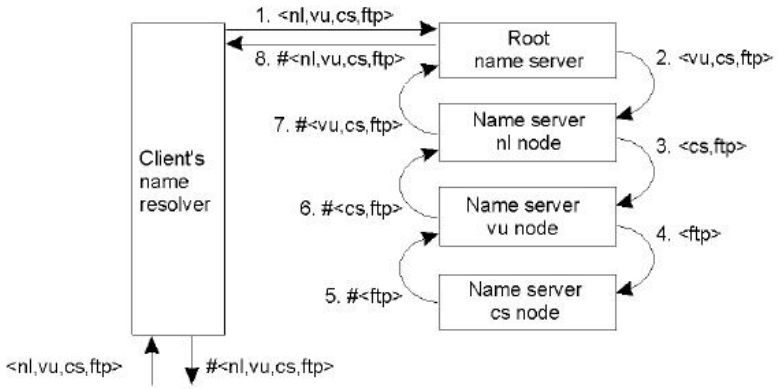


Erklären Sie verschiedene (hierarchische) Möglichkeiten der iterativen/rekursiven "name resolution".

ITERATIVE



REKURSIVE



29)
Erläutern Sie das Domain Name System DNS, sowie den Ablauf bei der Namens-Auflösung anhand der DNS Database (Resource Records). Was ist reverse lookup? Was ist ein zone-transfer?

Erläutern Sie das Domain Name System DNS, sowie den Ablauf bei der Namens-Auflösung anhand der DNS Database (Resource Records).

- DNS ist das Namenssystem für das Internet.
- Es wird verwendet um Domainnamen auf IP-Adressen zu mappen
- Eine Domain kann man sich als UNTERBAUM eines hierachischen Graphen vorstellen

Knoten speichern Ihre Daten in Resource Records. Diese enthalten unter anderem folgende Informationen

- SOA -> Start of Authority (Information zur dargestellten Zone)
- A -> Host (IP-Adresse)
- MX -> Mailserver (Mailserver, der Emails an diesen Knoten verarbeitet)
- NS -> Verweist auf einen Nameserver, der die dargestellte Zone implementiert
- CNAME -> Symbolischer Link auf den Primärnamen des dargestellten Knotens
- PTR -> kanonischer Name des Hosts (für reverse Lookup)

Was ist reverse lookup?

Wird benötigt um von einer IP-Adresse auf einen Domainnamen zu schließen.

zB: 193.168.2.1 -> www.beispiel.at

Was ist ein Zone-Transfer?

- Begriff wird im Rahmen der Replikation von Domain Name Server verwendet.
- Ist das Übertragen von Resource-Records von einem Primary Server auf einen Secondary Server
- Dient der Ausfallsicherheit und Performance

30)
Was ist Directory Service bzw. "Attribute-based naming"? Beschreiben Sie den prinzipiellen Aufbau des X.500 Name Space sowie dessen LDAP Implementierung.

Was ist Directory Service bzw. "Attribute-based naming"?

- Attributbasierte Namensgebung
- Bietet einen Suchdienst nach Entitäten über deren Attribute
- Suche einer Entität über (Attribut,Wert)-Paare.
- zB Wildcards suche

Beschreiben Sie den prinzipiellen Aufbau des X.500 Name Space sowie dessen LDAP Implementierung.

- Ein ehrgeiziger Versuch Informationen über das weltweite Informationssysteme zu kompilieren
- Jeder Knoten enthält mehrere (Attribut,Wert)-Paare
- Jeder Knoten hat einen eindeutigen Namen, der eine Kombination aus Namensattributen darstellt. Jedes Attribut wird als RDN (Relative Distinguished Name) bezeichnet.

LDAP

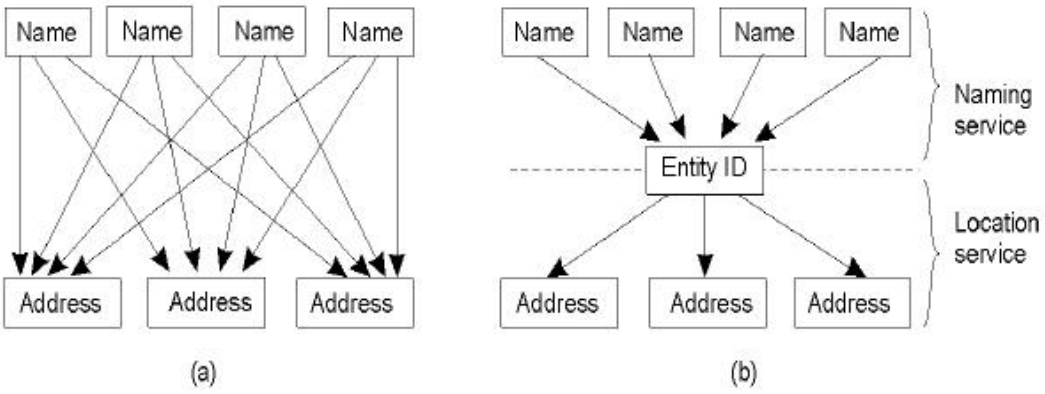
- DAP war zu schwer (vorgänger)
- **L**eightweight **D**irectory **A**ccess **P**rotocol (LDAP)
- Wurde aus X.500 Verzeichnisdienst abgeleitet.
- Ist eine effizientere, einfacherer Implementierung.
- Basiert auf TCP

31)
Wie funktioniert in flachen Namensräumen das Location Service? Geben Sie das Grundprinzip möglicher Lösungen an und gehen Sie dabei auch auf die Begriffe Mobility und Discovery ein. Erläutern Sie Vor- und Nachteile bei der Verwendung von "Forewarding Pointers". Wie funktionieren "Home-based approaches" für mobile Geräte?

Wie funktioniert in flachen Namensräumen das Location Service?

- Ein Location Service dient dazu, Services zu registrieren und aufzufinden.
- Beispielsweise bei P2P netzen, müssen Rechner Services und Ressourcen ankündigen bzw. Ressourcen anderer Rechner auffinden können.
- Abgrenzung zu Discovery Services: Bei Location Services ändern sich die Anforderungen oft und schnell.

Geben Sie das Grundprinzip möglicher Lösungen an und gehen Sie dabei auch auf die Begriffe Mobility und Discovery ein.



a) Direct, single level mapping between names and addresses.

b) Two-level mapping (e.g., using identities). Location service maps identifier to address.

- Direktes Mapping von Namen auf Adressen
- 2-Stufen-Mapping von Namen über Bezeichner (Identifier) auf Adressen

MOBILITY

Durch Location Services wird ein hoher Grad ein Mobilität erreicht, da ein Host nicht mehr über eine fixe Adresse (IP) angesprochen wird.

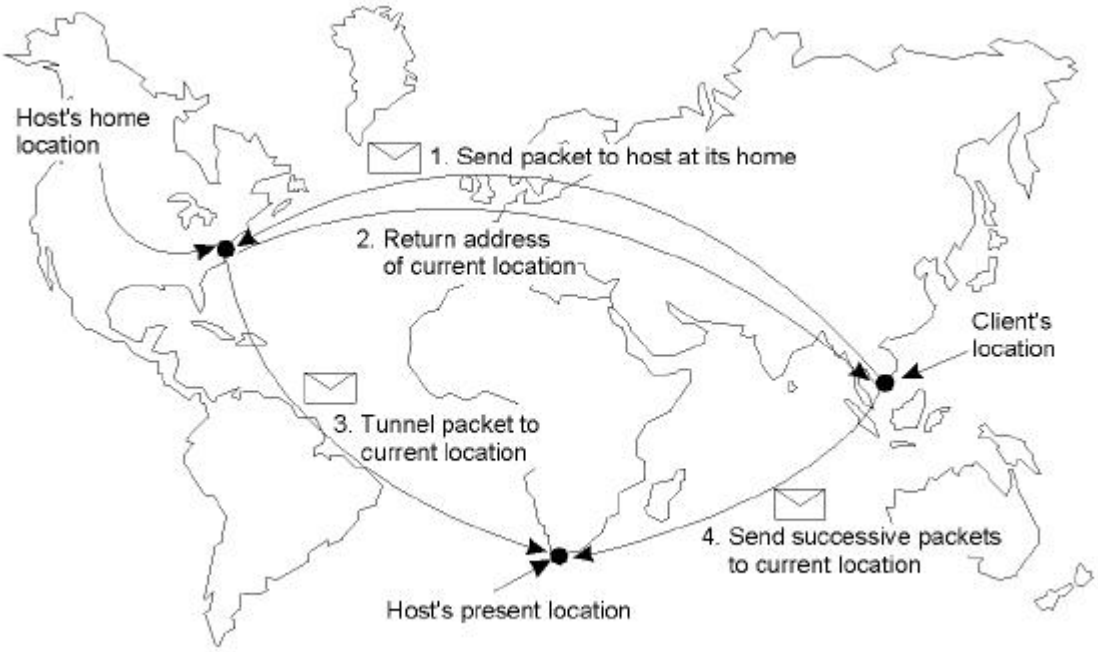
DISCOVERY

?

Erläutern Sie Vor- und Nachteile bei der Verwendung von "Forewarding Pointers".

- Wenn eine Entität seinen Ort verändert, lässt sie eine Referenz zurück, die auf ihren neuen Standort verweist.
- Probleme:
  -> Lange Ketten
  -> möglicherweise viele Zwischenschritte notwendig
  -> Eine falsche Referenz unterbricht die Verbindung (gebrochener Link)
- Ketten müssen so kurz wie möglich gehalten werden.

Wie funktionieren "Home-based approaches" für mobile Geräte?



- Der Heimatagent (Home-Agent) speichert immer den aktuellen Standort einer Entität
- Ändert eine mobiles Gerät (Entität) den Standort, so registriert/aktualisiert sie eine CARE-OF-ADDRESS beim Home-Agenten.

32)
Wozu braucht man Uhrensynchronisation? Erläutern Sie das NTP und den Berkeley Algorithmus. Was ist die Problematik bei der Synchronisation von Physical Clocks?

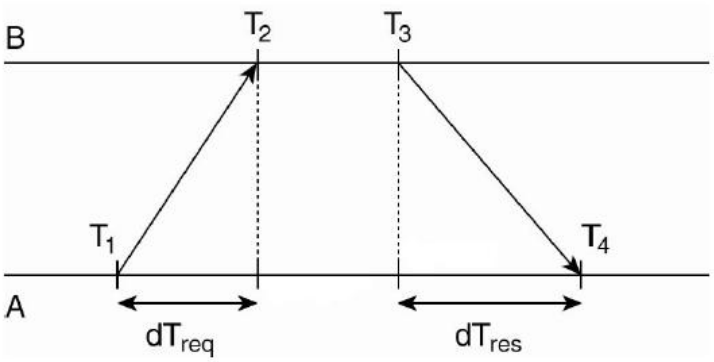
Wozu braucht man Uhrensynchronisation?

- Es ist nicht trivial in verteilten Systemen eine EINIGKEIT über die Zeit zu erzielen (mehrere physikalische Uhren in Computern, Kristalle schwingen Unterschiedlich)
- Uhrensynchronisation braucht man für die Festlegung der Reihenfolge von Operationen in verteilten Systemen
- Es geht um die Korrektheit verteilter, zeitabhängiger Informationsverarbeitung
- Es muss nicht immer eine „wirkliche“ Uhr verwendet werden, kann auch nur eine logische Uhr verwendet werden (Zeitgeber)

Erläutern Sie das NTP und den Berkeley Algorithmus.

NTP

- Wird verwendet zur Synchronisation mit einem **EXTERNEN** Zeitgeber.
- Die Genauigkeit von Zeitgebern wird durch das sogenannte **STRATUM** definiert.
  -> je niedriger es ist (absolute Wert), desto genauer ist die Zeitquelle
- Bei NTP wird mit jener Anfrage synchronisiert, die den niedrigsten Stratum-Wert hat.

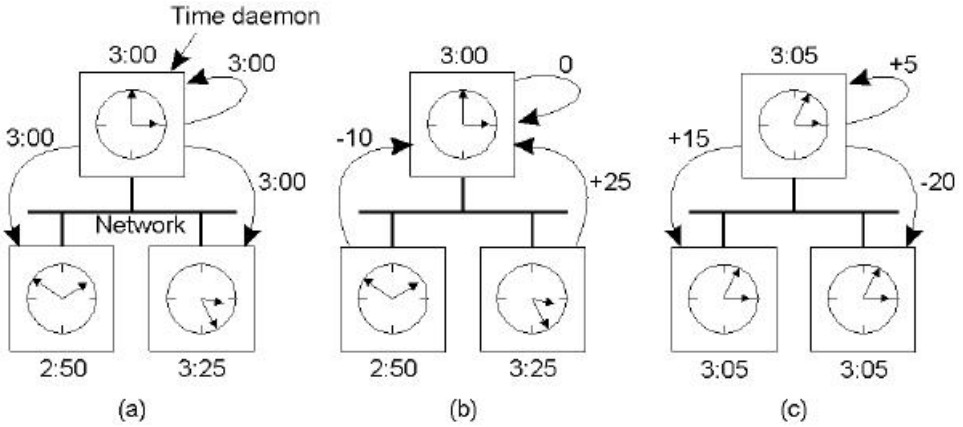


T1 (Sendezeitpunkt der Anfrage an B)
T2 (Empfangszeitpunkt bei B)
T3 (Sendezeitpunkt an A)
T4 (Empfangszeitpunkt der Anfrage bei A)

- Die die Zeit von B zum Zeitpunkt T3 in die Nachricht verpackt wird und erst zum Zeitpunkt T4 empfangen wird, entsteht so eine Verzögerung (Delay).
- Es werden mehrere Anfragen gesendet und die Anfrage mit dem kleinesten Delay/STRATUM verwendet.
- Man kann die Übertragungszeit durch Berechnung korrigieren
- Die Uhr wird dann langsam vorgestellt, jedoch NIEMALS zurückgestellt
  -> zurückstellen ist problematisch wegen Zeitstempel. Zeitstempel müssen immer linear vorwärts gehen.
- STRATUM darf auch nicht über einem gewissen Grenzwert liegen, da ansonsten Zeitquelle nicht vertrauenswürdig ist.

BERKELEY ALGORITHMUS

- Dabei handelt es sich um eine interne Synchronisation
- Es gib einen Koordinator (Time daemon), der seine lokale Zeit an alle sendet (auch an sich selbst)
- Jeder antwortet mit seiner Abweichung von dieser Referenzzeit.
- Danach berechnet der Koordinator die durchschnittliche Abweichung und sendet sie jedem zurück.
- Diese beschleunigen oder verlangsamen dann ihre Zeit entsprechend der Korrektur.



a) The time daemon asks all the other machines for their clock values
b) The machines answer
c) The time daemon tells everyone how to adjust their clock

Was ist die Problematik bei der Synchronisation von Physical Clocks?

- Uhren dürfen NIE zurückgestellt werden, nur über längere Zeit verlangsamt werden
- Antwortzeiten über Netzwerk unterschiedlich
- Unterschiedliche Schwingfrequenzen der Kristalle
- Wie werden die einzelnen Uhren mit Uhren der Realität synchronisiert
- Wie können die einzelnen Uhren untereinander synchronisiert werden

33)
Was sind die Gründe für die Verwendung von Logical Clocks? Erklären Sie die Unterschiede zu den Physical Clocks. Was ist die "happened-before" Beziehung und wie funktionieren die "Lamport-Timestamps"?

Was sind die Gründe für die Verwendung von Logical Clocks?

Bisher nahmen wir an, dass die Uhrensynchronisation von Natur aus mit der realen Zeit zusammenhängt. Wir haben jedoch auch gesehen, dass es ausreichen kann, wenn sich alle Knoten auf eine aktuelle Zeit einigen, die nicht notwendigerweise mit der realen Zeit übereinstimmt.

Erklären Sie die Unterschiede zu den Physical Clocks.

- Physikalische Uhr ist die „reale“ Uhr (Kristalloszillator, mittlere Sonnensekunde, Atomuhr (TAI, international Atomic Time, UTC
- Eine logische Uhr ist nur für die Reihung der Ereignisse relevant (happens-before)

Was ist die "happened-before" Beziehung und wie funktionieren die "Lamport-Timestamps"?

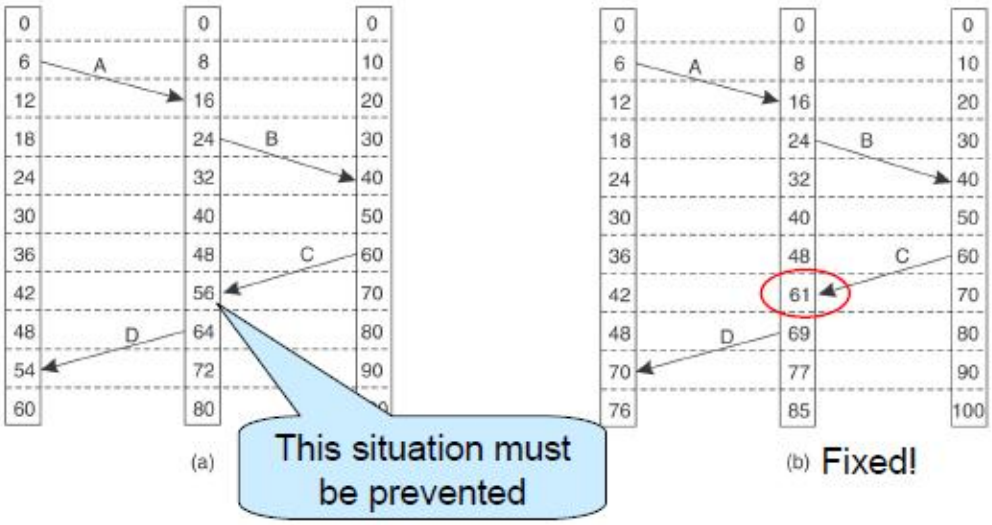
HAPPENED-BEFORE

a -> b wird als a passiert vor b gelesen.

Dies bedeutet, dass alle Prozesse wissen a->b

- Wenn a und b Ereignisse im selben Prozess sind und a vor b eintritt, dann ist a->b wahr
- Wenn a das Ereignis ist, dass eine Nachricht von einem Prozess ausgesendet wurde, und b das Ereignis, dass diese Nachricht von einem anderen Prozess empfangen wurde, dann ist a->b auch wahr.
- Eine Nachricht kann nicht empfangen werden, bevor sie gesendet wurde
- Diese Relation ist auch transitiv: a->b und b->c, dann gilt a->c

LAMPORT TIMESTAMPS



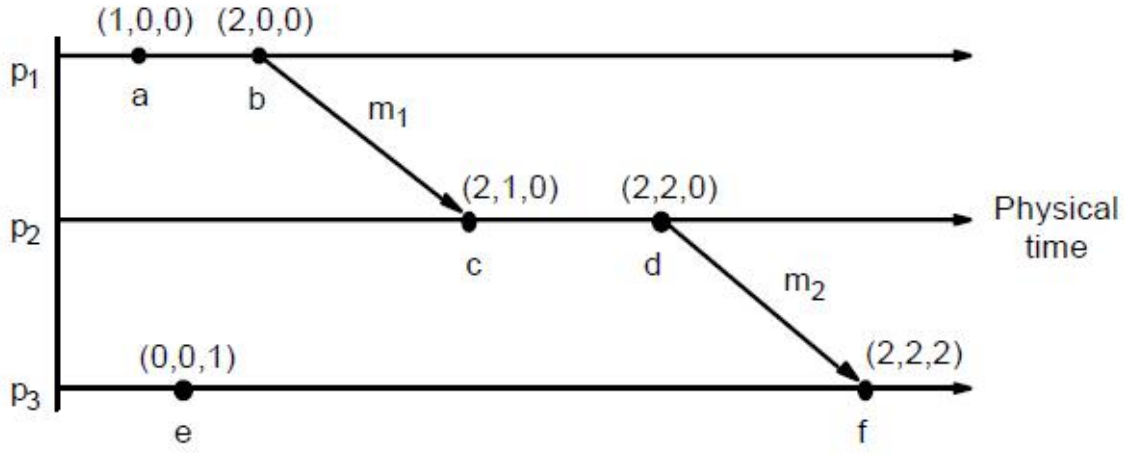
- Jeder Prozess führt seine eigene Uhr (Clock, Zähler)
- Dieser Zähler wird bei jedem Ereignis zum eins hochgezählt
- Beim Senden einer Nachricht wird er eigene aktuelle Zähler mitgesendet
- Es muss immer gelten T(b)>T(a) -> Zeit des Empfängers muss immer größer sein als Zeit des Senders
  -> ist dies nicht der Fall, wird die Zeit beim Empfänger korrigiert (siehe oben b)

34)
Welchen Nachteil haben die Lamport-Timestamps und wie kann dieser durch Vector-Timestamps überwunden werden?

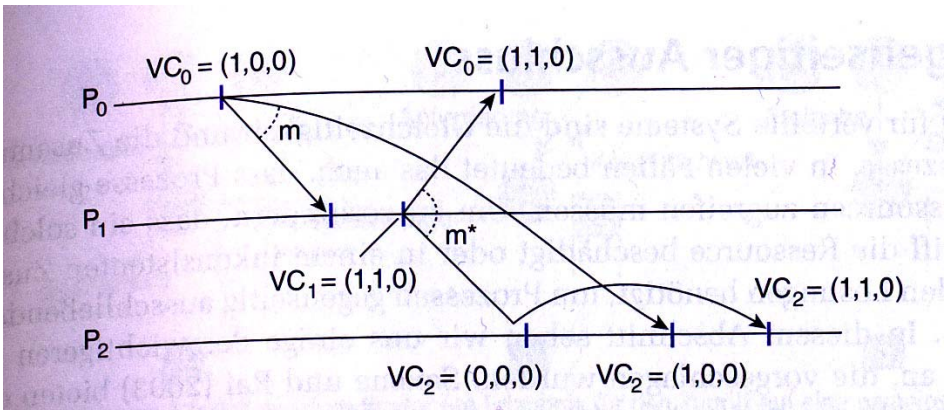
NACHTEIL VON LAMPORT TIMESTAMPS

Lamport Timestamps sagen nichts über kausale Abhängigkeiten aus.

VECTOR TIMESTAMPS



- Jeder Prozess führt einen Vektor mit den jeweiligen logischen Uhren der anderen Prozesse
- Bei jedem Ereignis im Prozess i zählt dieser sein Element i um eins hoch
- Beim Empfang der Nachricht durch Prozess j wird das Weitergeben dieser an die darübergestellte Anwendung solange verzögert bis folgende Beziehung gilt:
  -> $ts(m)[i] = VC_j[i]+1$
  -> $ts(m)[k] \leq VC_j[k]$ für alle $k \neq i$



35)
Wie funktioniert Distributed Mutual Exclusion. Wie verhalten sich verschiedene Algorithmen (centralized, distributed, token-ring) hinsichtlich Skalierbarkeit und Fehlertoleranz

---

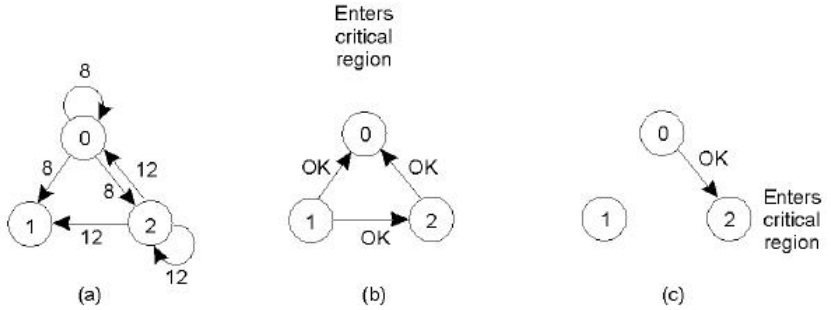
Wie funktioniert Distributed Mutual Exclusion.

Das Ziel von Distributed Mutual Exclusion ist, in einem verteilten System den **EXKLUSIVEN ZUGRIFF** auf eine Ressource zu vergeben. Es soll verhindert werden, dass 2 Prozesse gleichzeitig auf dieselbe Ressource zugreifen.

Wie verhalten sich verschiedene Algorithmen (centralized, distributed, token-ring) hinsichtlich Skalierbarkeit und Fehlertoleranz

- **CENTRALIZED**
  - Ein Prozess agiert als Koordinator
  - Koordinator vergibt exklusiven Zugriff auf eine Ressource
  - Bei mehreren Anfragen fungiert er als Warteschlange
  - Starvation ist nicht möglich
  - **PROBLEM:** Wenn Koordinator ausfällt, gibt es keinen Zugriff mehr.

- **DISTRIBUTED**



  - Wenn ein Prozess Zugriff auf eine Ressource will, sendet er an alle anderen beteiligten Prozessen einschließlich sich selbst ein REQUEST.
  - Beim Empfangen dieser Nachricht gibt es 3 mögliche Antworten:
  * Greift er selbst auf die Ressource zu, antwortet er mit DENIED
  * Greift er nicht darauf zu, antwortet er mit OK
  * Greift er nicht darauf zu, will es aber tun, so vergleicht er seine eigene CLOCK mit jener der CLOCK der Anfrage (wird mitgeschickt). Die frühere (niedrigere) gewinnt. Je nachdem wird mit OK oder DENIED geantwortet.
  - **PROBLEM:** Wenn einer der Prozesse abstürzt kommt es zu einem Crash, da auf die Antwort aller gewartet wird.

- **TOKEN-RING**
  - Ein Token wird zwischen den Prozessen weitergereicht
  - Nur jener Prozess, der das Token hält ist zum Zugriff berechtigt
  - Benötigt ein Prozess keinen Zugriff, so reicht er das Token weiter
  - **PROBLEM:** Wenn Prozess der Token hält abstürzt, ist es schwierig dies zu erkennen
  - **PROBLEM:** Token muss immer weitergereicht werden, auch wenn aktuell kein Zugriff auf eine Ressource benötigt wird

36)
Sie sollen einen Dateiserver implementieren der mehrere Clients gleichzeitig bedienen kann. Nennen und beschreiben Sie einen (konkreten) Mechanismus der garantiert dass immer nur einer der Clients gleichzeitig eine Datei schreiben darf. Die anderen Clients sollen solange blockiert werden (egal ob schreibend oder lesend) bis der schreibende Client seine Arbeit beendet hat. Warum ist das sinnvoll?

CENTRALIZED Algorithmus

- Ein zentraler Koordinator nimmt die READ und WRITE Anforderungen der Clients entgegen.
- READ dürfen parallel von mehreren Clients durchgeführt werden.
- WRITE darf nur ein einzelner CLIENT durchführen.
  -> sobald WRITE freigegeben wird, darf kein READ mehr durchgeführt werden. Alle READ-Requests kommen in eine Warteschlange. Sobald WRITE-Request fertig gestellt ist, werden weitere WRITE-Anfragen in der Warteschlange bearbeitet, ansonsten werden die anderen READ-Requests bearbeitet.
- WRITE –Requests kommen ebenso in eine Warteschlange.

SINNVOLL:

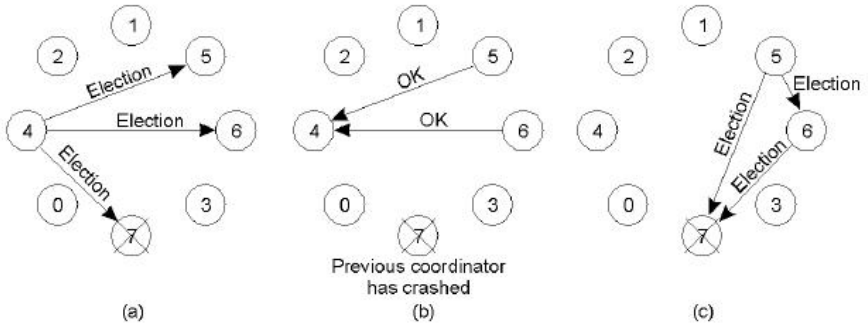
- Wenn ein Schreibvorgang noch nicht beendet ist und ein READ ausgeführt wird, dann erhält der Client eine ungültige Datei.
- Ein Schreibvorgang darf nicht in einen anderen Schreibvorgang „hineinschreiben“. Würden sich gegenseitig die Datei zerstören.

37)
Vergleichen Sie den "Bully" und den "Ring"-Algorithmus für Election hinsichtlich Fehlertoleranz. Warum sind diese Algorithmen für ad-hoc oder large-scale Systeme weniger geeignet und welche grundsätzlichen Lösungsansätze verfolgt man daher dort?

Vergleichen Sie den "Bully" und den "Ring"-Algorithmus für Election hinsichtlich Fehlertoleranz.

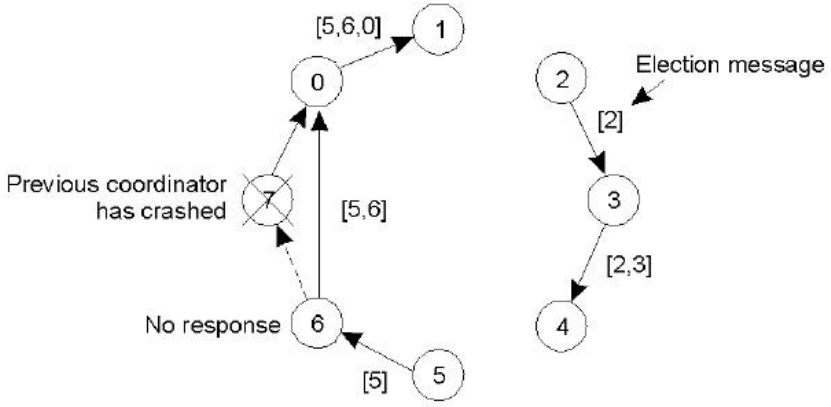
BULLY ALGORITHMUS

- Wenn der Koordinator ausgefallen ist und es ein Prozess bemerkt, sendet er eine ELECTION-Nachricht an alle Prozesse mit einer höheren ID.
- Wenn er keine Antwort erhält, so ist er neuer Koordinator (informiert dann alle anderen Prozesse, dass er neuer Koordinator ist)
- Antwortet einer der anderen Prozesse, so übernehmen diese die **KONTROLLE**, wer neuer Koordinator wird.
- Dieser wiederholt den Vorgang, bis der Prozess mit der höchsten ID bekannt ist und eine COORDINATOR-Nachricht aussendet.



RING ALGORITHMUS

- Die Prozesse sind logisch in Form eines Rings angeordnet.
- Bemerkt ein Prozess, dass ein Koordinator ausgefallen ist, so wird eine ELECTION-Nachricht an seinen Nachfolger gesendet. Wenn dieser Antwortet ist das OK und es ist erledigt. Ansonsten wird an den übernächsten die Nachricht gesendet, bis eine Antwort OK kommt.
- Jeder Prozess hängt seine eigene ID an.
- Die Nachricht zirkuliert solange, bis sie wieder den Ausgangsprozess erreicht.
- Dieser prüft die Einträge und jener mit der höchsten ID wird zum Koordinator gewählt.
- Damit informiert der Prozess alle anderen über die ID des neuen Koordinator (COORDINATOR-Nachricht)
- Eine Wahl kann von mehreren Prozessen gestartet werden, ändert nichts am Ergebnis.



VERGLEICH hinsichtlich FEHLERTOLERANZ

Beide Algorithmen sind sehr Fehlertolerant, da immer wieder ein neuer Koordinator gewählt werden kann. Es besteht KEIN Single-Point-Failure Problem.

Warum sind diese Algorithmen für ad-hoc oder large-scale Systeme weniger geeignet und welche grundsätzlichen Lösungsansätze verfolgt man daher dort?

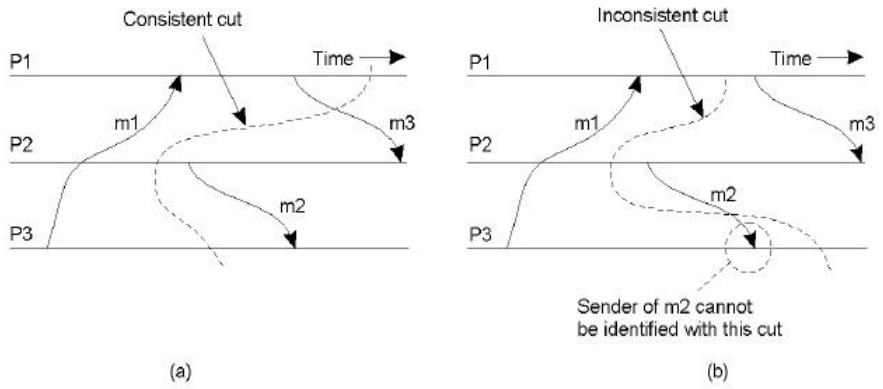
AD-HOC SYSTEME

Bei ad-hoc Systemen ist es wichtig, dass nicht irgendein Koordinator gewählt wird, sondern der Beste anhand verschiedener Kriterien wie:
- Bandbreite

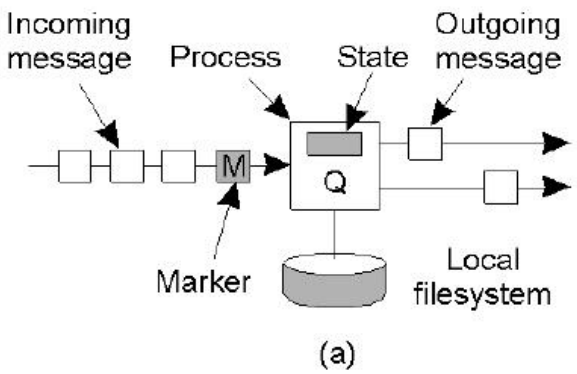
LARGE-SCALE-NETZWERKE

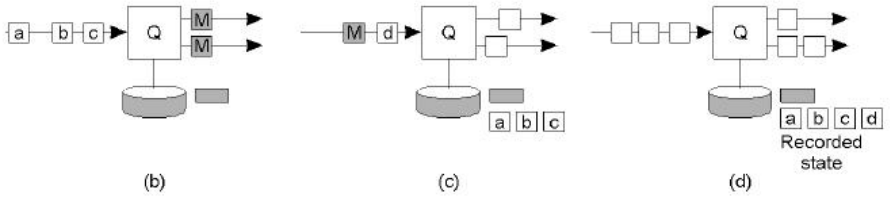
- Auswahl von Superpeers als Koordinatoren mit niedriger Latenzzeit.
- Jeder Superpeer sollte nicht mehr als einer fest vorgegebenen Anzahl an Knoten dienen
- Superpeers sollen möglichst gleichmäßig über das Overlay-Netzwerk verteilt werden.
- Die Anzahl der Superpeers soll relativ zu der gesamten Anzahl an Knoten sein.

38)
Welche Probleme gibt es bei der Ermittlung des "Global State" und wie können diese überwunden werden? Geben Sie zumindest einen Algorithmus an.



a) A consistent cut
b) An inconsistent cut (effect without cause)





b) Process Q receives a marker for the first time (from other channel) and records its local state
c) Q records all incoming message
d) Q receives a marker for its incoming channel and finishes recording the state of the incoming channel

Algorithmus von Chandy und Lamport in den Folien/Buch nachschlagen (Frage noch genauer Ausarbeiten).

39)
Was sind die Hauptgründe für den Einsatz von Replikation in verteilten Systemen? In welcher Beziehung stehen Replikation und Skalierbarkeit zueinander? Erläutern Sie in diesem Zusammenhang verschiedene Varianten der Content Replication und des Content Placement.

Was sind die Hauptgründe für den Einsatz von Replikation in verteilten Systemen?

Bei Replikation führt man mehrere Kopien einer Entität auf verschiedenen Knoten.

- **Fehlertoleranz (Redundanz)**
  -> Umschalten im Fehlerfall
  -> Schutz gegen beschädigten Daten (corrupted data)
- **Performance**
- **Skalierbarkeit**
  -> Skalierung in der Anzahl der Knoten/Replikate
  -> Skalierung in geographischem/topologischem Kontext

In welcher Beziehung stehen Replikation und Skalierbarkeit zueinander?

- Skalierbarkeit und Replikation stehen insofern in der Beziehung, dass die Replikation einen deutlichen Mehraufwand bedeutet, da die Daten konsistent gehalten werden müssen.
- Man muss abschätzen, ob das Einsetzen von Replikaten in einem Verhältnis steht zum Mehraufwand an Traffic, Ressourcen, Zugriffszeit
-

Erläutern Sie in diesem Zusammenhang verschiedene Varianten der Content Replication und des Content Placement.

?

40)
Geben Sie verschiedene Möglichkeiten der Update Propagation (Content Distribution) an und bewerten Sie diese hinsichtlich Vor- und Nachteilen sowie Einsatzmöglichkeiten.

Fragestellung: Wann ist das Update und was wird zu den Replikaten transferiert.

STATE vs. OPERATION

- **Notification (Benachrichtigung)**
  -> Weitergabe der Aktualisierung nur durch eine Benachrichtigung über die geänderten Daten (Daten werden für ungültig erklärt)
  -> benötigt wenig Bandbreite
  -> gut wenn viele Aktualisierungsoperation im Vergleich zu Leseoperationen sind
- **Operation transfer (Weitergabe der Aktualisierungsoperation)**
  -> hier werden keine Änderungen der Daten übertragen, sondern lediglich die Operation, die dann jedes Replikat lokal ausführt.
  -> Vorteil ist, dass Aktualisierungen im Bezug auf Verwendung minimaler Bandbreite weitergegeben werden
  -> Nachteil: Jedes Replik benötigt womöglich mehr Rechenleistung, insbesondere wenn die Operationen realtiv komplex ist.
- **State transfer (Übertragung der Aktualisierungs-Daten)**
  -> Übertragung der vollen geänderten Daten
  -> Sinnvoll dort, wo ein hohes Lese-/Schreibverhältnis

PUSH-SERVER BASED PROTOKOLL

- Im Push-basierten Ansatz werden Aktualisierungen an andere Repliken weitergegeben, ohne dass diese welche angefordert haben
- Werden dort eingesetzt, wo die Replike im allgemeinen einen relativ hohen Konsistenzgrad aufweisen müssen
- Der Server muss alle Clients verwalten, die von ihm Aktualisierungen erhalten

PULL-CLIENT BASED PROTOKOLL

- Im Pull basierten Ansatz fordert ein Server oder ein Client einen anderen Server auf, ihm Aktualisierungen zu schicken, die ihm zu diesem Zeitpunkt vorliegen.
- Wird oft von Clients verwendet, wo der Browser zuerst prüft, ob sein Cache noch aktuell ist, und dann bei Bedarf die Aktualisierung vom Webserver ladet.

HYBRID: LEASE

- Vereinbarung zwischen Client und Server, dass der Server regelmäßig updates für eine fixe Zeit sendet.
- Nach Ablauf fordert der Client eine neue Vereinbarung an.

UNICAST vs. MULTICAST

- UNICAST
  -> Bei Unicast Kommunikation sendet ein Server, der Teil des Datenspeichers ist, seine Aktualisierung an N andere Server, indem er N separate Nachrichten sendet.
  -> wird häufig bei einem pull-basiertem Ansatz verwendet, da nur ein einzelnen Client ist, der eine Aktualisierung anfordert.
- MULTICAST
  -> Beim Multicast übernimmt das zugrundeliegende Netzwerk die Aufgabe, eine Nachricht effizient an mehrere Empfänger zu senden.
  -> wird häufig effizient mit einem push-basierten Ansatz kombiniert, um Aktualisierungen weiterzugeben

41)
Erläutern Sie die Funktionsweise der "primary-based" Protokolle. Bewerten und vergleichen Sie die verschiedenen Arten.

Urbildbasierte Protokolle (Replikationsprotokolle)

- Primary-based Protokolle erlauben den Schreibzugriff nur auf einem Replikat.
- Ein Schreibrequest auf einem beliebigen anderen Replikat wird an den Primary-Server weitergereicht (remote-write).
- Der Primary-Server kann dabei für jedes Datenelement ein anderer Host sein

PROTOKOLLE FÜR ENTFERNTE SCHREIBVORGÄNGE

Ein Prozess der eine Schreib-Operation an dem Datenelement x vornehmen will, leitet die Operation an den für x primären Server weiter. Dieser führt die Aktualisierung seiner lokalen Kopie durch und leitet sie anschließend an die Backupserver weiter. Jeder Backupserver seinerseits für die Aktualisierung seiner lokalen Kopie aus und gibt dem primären Server eine Bestätigung zurück.

PROTOKOLLE FOR LOKALE SCHREIBVORGÄNGE

Bei dieser Variante MIGRIERT die primäre Kopie zwischen den Prozessen, die eine Schreiboperation vornehmen wollen. Dabei macht der Prozess die primäre Kopie von x ausfindig, VERSCHIEBT sie an seinen eigenen Standort und ändert sie lokal.

Es können so mehrere Schreiboperationen nacheinander lokal erfolgen.

42)
Erläutern Sie die Funktionsweise der "replicated-write" Protokolle. Bewerten und vergleichen Sie die verschiedenen Arten. Welche Probleme können bei "Active Replication" auftreten? Erklären Sie "quorum-based" Replikationsprotokolle und geben Sie die Bedingungen an, um Read-Write und Write-Write Konflikte zu verhindern.

Erläutern Sie die Funktionsweise der "replicated-write" Protokolle. Bewerten und vergleichen Sie die verschiedenen Arten.

Schreiboperationen können nicht nur auf einem Replikat stattfinden, sondern auf mehreren Replikaten.

AKTIVE REPLIKATION

- Ähnelt sehr der urbildbasierten Protokolle
- Jedem Replikat ist ein Prozess zugeordnet, der Aktualisierungsoperationen vornimmt.

Welche Probleme können bei "Active Replication" auftreten?

- Operationen müssen überall in derselben Reihenfolge ausgeführt werden.
- Es wird ein auf absoluter Ordnung basierender Multicasting-Mechanismus benötigt.
- Lässt sich über die Lamport logischen Uhren erreichen

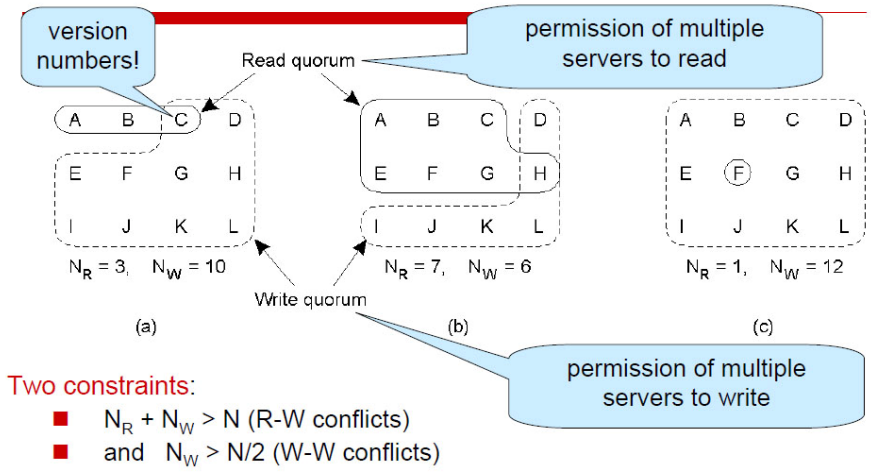
QUORUMGESTÜTZTE REPLIKATION

- Basiert auf Abstimmung
- Der Grundgedanke ist, dass Clients zuerst die Erlaubnis mehrere Server einholen müssen, bevor sie ein repliziertes Datenelement lesen oder schreiben dürfen

Erklären Sie "quorum-based" Replikationsprotokolle und geben Sie die Bedingungen an, um Read-Write und Write-Write Konflikte zu verhindern.

Beispiel:

Ein Client will eine Datei lesen, die auf N Server repliziert wird. Es gibt nun eine Regel die gesagt, dass der Client zuerst eine Mehrheit, dh N/2 +1 Server befragen muss und eine Zustimmung einholen muss bevor er eine Datei schreiben kann. Auch beim Lesevorgang gilt dasselbe.



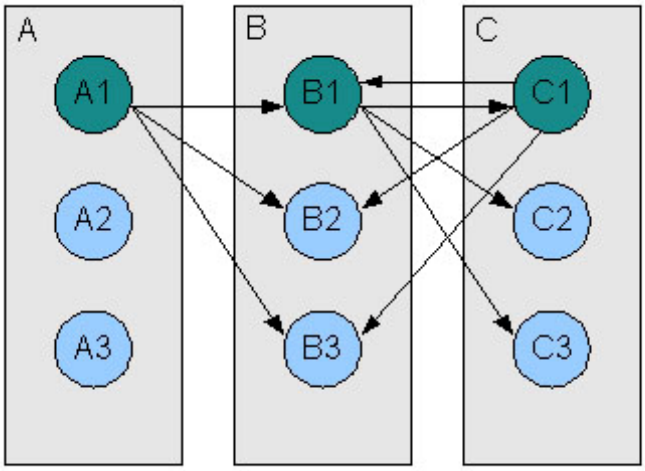
Mit den gegebenen Einschränkungen werden Lese-Schreib-Konflikte sowie Schreib-Schreib-Konflikte vermieden.

COORDINATOR-COHORT-REPLICATION

- Eine Spezialform von Replicated-Write-Protokollen.
- Dabei wird die Anfrage an ein (beliebiges) Replikat gesendet, das die Updates synchron an alle anderen propagiert.
- Dazu ist ein DISTRIBUTED-LOCKING-MECHANISMUS erforderlich

43)
Welche Besonderheiten sind bei der Replikation von Objekten zu beachten? Erläutern Sie (inkl. genauer Skizze) wie man "Replication transparency" in Objektsystemen umsetzen könnte ("replicated invocation").

- Eine Menge von Replikaten wird zu einer Gruppe zusammengefasst
- Ein einzelnes Replikat ist Koordinator dieser Gruppe
- Bei Aktualisierung wird immer eine Methode aller Replikate aufgerufen
- Möchte jedoch ein Objekt das selbst gerade aktualisiert wird, wiederum eine Aktualisierung eines weiteren ENTFERNTEN Objekts aufrufen, ist dies nicht möglich, da ansonsten diese Methode ZU OFT aufgerufen wird.



Falls als A ein Aufruf in B erfolgt, so ruf A alle Replikate in B auf (also B1, B2, B3). Wenn B nun seinerseits wieder eine Methode von C aufruft, so darf dies wieder nur vom Koordinator von B (B1) erfolgen. B1 ruft bei allen Replikaten von C (C1, C2, C3) die Methode auf.

Dies darf nur der Koordinator machen, die die Methode von C ansonsten zu oft aufgerufen wird.

Die Antwort darf ebenso nur vom Koordinator zurückgegeben werden.

Dh nur C1 gibt in dem Beispiel die Antwort an **ALLE** Replikate zurück.

44)
Was sind Epidemic Protocols. Welche Vor- und Nachteile haben diese? Erklären Sie „gossiping“ („rumor spreading“) im Zusammenhang mit Replica update propagation. Erläutern Sie Vor- und Nachteile. Erklären Sie das Anti-Entropy Modell im Zusammenhang mit Replica update propagation. Erläutern Sie Vor- und Nachteile.

Was sind Epidemic Protocols. Welche Vor- und Nachteile haben diese?

- Sind für Datenspeicher gedacht, welche nur eine eventuelle Konsistenz aufweisen müssen
  -> dh wenn es keine Aktualisierung gibt, muss nur sichergestellt sein, dass alle Repliken irgendwann identisch sind.

Vorteile:

- Gute Skalierbarkeit
- Aktualisierung an alle Repliken erfolgt in so wenigen Nachrichten wie möglich (Netzbelastung)

Nachteile:

- Weitergabe des Löschens eines Datenelements ist schwierig
- Löst keine Aktualisierungskonflikte

Erklären Sie "gossiping" ("rumor spreading") im Zusammenhang mit Replica update propagation. Erläutern Sie Vor- und Nachteile.

Prinzip der Infizierung

- Ist ein effizientes Weitergabemodell für epidemische Protokolle
- Wenn ein Datenelement aktualisiert wurde, wendet sich der Server an einen beliebigen anderen Server um die Aktualisierung weiterzugeben usw.
- Wenn ein Server schon infiziert wurde, so macht er nur mit einer gewissen Wahrscheinlichkeit weiter.
- Es kann nicht garantiert werden, dass alle Server infiziert werden

Erklären Sie das Anti-Entropy Modell im Zusammenhang mit Replica update propagation. Erläutern Sie Vor- und Nachteile.

ANTI-ENTROPY

Ist ein Weitergabemodell für epidemische Protokolle, welches per Zufall einen anderen Server auswählt.

45)
Erläutern Sie die grundlegenden Begriffe der Dependability. Nennen Sie die fünf wesentlichen Attribute (bzw. Requirements) eines "dependable system". Was ist der Unterschied zwischen Availability und Reliability? Erläutern Sie die "dependability threats" Failure, Error und Fault sowie den Zusammenhang zwischen den drei. Erläutern Sie "permanent", "transient" und "intermittent" faults anhand von Beispielen.

Erläutern Sie die grundlegenden Begriffe der Dependability.

- Ist die Fähigkeit eines Systems einen Service zur Verfügung zu stellen, dem GERECHTFERTIGTERWEISE vertraut werden kann
- Ist die Eigenschaft eines Systems, dass es regelmäßige und gravierende Systemausfälle vermeidet, sodass der Dienst als verlässlich gilt und es für den Benutzer akzeptabel ist.
- Bedeutet, dass es auch im Fehlerfall seinen Dienst zur Verfügung stellt
- Ein Fehlerausfall beeinträchtigt nur einen Teil des Systems, jedoch nicht das komplette System.

Nennen Sie die fünf wesentlichen Attribute (bzw. Requirements) eines „dependable system“.

- **AVAILABILITY/VERFÜGBARKEIT**
  ->Verfügbarkeit ist definiert als die Eigenschaft, dass ein System unnmittelbar benutzbar ist.
  - >Im Allgemeinen bezeichnet sie die Wahrscheinlichkeit, dass das System zu einem bestimmten **Zeitpunkt** verfügbar ist.
- **RELIABILITY/ZUVERLÄSSLICHKEIT**
  -> Bezeichnet die Eigenschaft, dass ein System fortlaufend ausfallfrei läuft.
  -> Im Gegensatz für Verfügbarkeit ist die Zuverlässigkeit für ein **Zeitintervall** definiert, anstatt für einen bestimmten Zeitpunkt.
  -> Wie lange (Zeitspanne) ein System ohne Ausfall arbeiten kann.
- **SAEFTY/SICHERHEIT**
  -> bezeichnet den Umstand, dass keine Katastrophe eintritt, wenn ein System vorübergehend nicht korrekt arbeitet.
  -> Beispielsweise müssen Steuerungen für Kernkraftwerke ein hohes Maß an Funktionssicherheit bieten.
- **INTEGRITY/INTEGRITÄT**
  -> Das System soll nicht in einen ungewünschten/nicht definierten Zustand gelangen.
- **MAINTAINABILITY/WARTBARKEIT**
  -> bezieht sich darauf, wie leicht ein ausgefallenen System repariert werden kann
  -> Hohe Wartbarkeit kann zu einer hohen Verfügbarkeit führen, da im Fehlerfall automatisch repariert wird.

Was ist der Unterschied zwischen Availability und Reliability?

- Availability ist die Wahrscheinlichkeit, dass ein System zu einem bestimmten Zeitpunkt verfügbar ist.
- Reliability ist für ein Zeitintervall definiert, im welchen das System ohne Fehler arbeitet.

Erläutern Sie die "dependability threats" Failure, Error und Fault sowie den Zusammenhang zwischen den drei.

- **Failure:** Ein Failure tritt ein, wenn ein System durch einen Error seine Dienste nicht mehr vollständig oder überhaupt nicht mehr zur Verfügung stellt.
- **Error:** Ein Teil des System Status, welcher durch einen Fault ausgelöst wurde und schlussendlich zu einem failure führt.
- **Fault:** ist die Ursache eines Fehlers

Erläutern Sie "permanent", "transient" und "intermittent" faults anhand von Beispielen.

PERMANENT FAULT

Treten solange dauerhaft auf, bis die fehlerhafte Komponente ersetzt wurde.
Bsp.: Hardwaredefekte, Software Bugs

TRANSIENT FAULT

Treten zufällig jedoch nicht wiederholt auf.
Bsp.: Ein Vogelschwarm stört die Funkverbindung

INTERMITTENT FAULT

Treten zufällig auf, verschwinden wieder und kommen dann wieder.
Diese Faults sind besonders schwer zu identifizieren
Bsp.: Ein loser Netzwerkstecker, welcher zu einem Wackelkontakt führt.

46)
Geben Sie verschiedene Fehlermodelle ("failure models") an und diskutieren Sie diese. Wozu benötigt man überhaupt Fehlermodelle? Inwiefern ist es u.U. heikel zu spezifizieren, daß ein System "k-fault-tolerant" sein soll?

Geben Sie verschiedene Fehlermodelle ("failure models") an und diskutieren Sie diese.

ABSTURZAUSFALL (Crash Failure)

Ein Server steht, hat aber bis dahin richtig gearbeitet. Der angebotene Dienst bleibt beständig aus (ständiger Dienstausfall)

DIENSTAUSFALL (Omission Failure)

- Der Server antwortet nicht auf eingehende Anforderungen
- Ein Server erhält keine eingehenden Anforderungen
- Ein Server sendet keine Nachrichten
  -> das kann verschiedene Gründe haben, beispielsweise dass er nie die Anfrage erhalten hat (receive omission) oder aber diese erhalten hat, auch bearbeitet aber nicht in der Lage ist zu antworten.

ZEITBEDINGTER AUSFALL (Timing Failure)

Wenn eine Antworte länger als eine definierte Response-Time benötigt, wird von einem Timing failure gesprochen.

AUSFALL KORREKTER ANTWORT (Response Failure)

- Die Antwort eines Servers ist falsch. Dieser Ausfall wird oft auch kurz Antwortfehler genannt.
- Der Wert der Antwort ist falsch.
- Der Server weicht vom richtigen Programmablauf ab.

BYZANTINISCHER ODER ZUFÄLLIGER AUSFALL (Arbitrary oder Byzantine Failure)

Der Server erstellt zufällige Antworten zu zufälligen Zeiten.

Wozu benötigt man überhaupt Fehlermodelle?

- Wenn ein Server von einem anderen Server abhängig ist, um seine Dienste bereitzustellen, muss die Fehlerursache manchmal auch an einer anderen Stelle gesucht werden.
- Solche Abhängigkeitsbeziehungen gibt es in verteilten Systemen im Überfluss
- Um besser zu verstehen, wie ernsthaft ein Ausfall wirklich ist, wurde ein KLASSIFIZIERUNGSVERFAHREN (Fehlermodelle) entwickelt.

Inwiefern ist es u.U. heikel zu spezifizieren, daß ein System "k-fault-tolerant" sein soll?

K-FAULT-TOLERANT

Heißt, in einem System können bis zu k-Komponenten Ausfallen, ohne dass das System davon beeinträchtigt wird (dh es liefert nach wie vor korrekte Ergebnisse)

Es ist insofern schwierig da, bei beispielsweise bei einem BYZANTINISCHEN Fehler, wo die fehlerhaften k-Prozesse sowohl korrekte als auch falsche Ergebnisse liefern es nötig ist, dass mindestens 2k+1 Prozesse nötig sind, damit mindestens k+1 Prozesse korrekt arbeiten.

Dann ist das System k-fault-tolerant.

47)
Wieso benötigt man Redundanz zur Maskierung von Fehlern? Welche Arten von Redundanz gibt es?

Wenn ein System fehlertolerant sein soll, kann es versuchen, das Auftreten von Ausfällen vor anderen Prozessen zu verbergen.

Schlüsseltechnik dazu ist die Verwendung von Redundanz.

INFORMATIONSREDUNDANZ
Dabei werden zusätzliche Bits hinzugefügt, um eine Wiederherstellung durcheinandergeratener Bits zu ermöglichen.

ZEITLICHE REDUNDANZ
Wird eine Aktion ausgeführt und dann, falls erforderlich wiederholt.
Transaktionen verwenden diesen Ansatz.
Wenn eine Transaktion abgebrochen wird, kann sie problemlos wiederholt werden.

TECHNISCHE REDUNDANZ
Dabei werden zusätzliche Ausrüstungen oder Prozesse hinzugefügt, damit das System als Ganzes den Verlust oder die Fehlfunktion bestimmter Komponenten kompensieren kann.

48)
Erläutern Sie die Aussage des "two-army" Problems.

Im **TWO-ARMY** Problem geht es darum, dass mit unzuverlässiger Kommunikation 2 Prozesse nicht zu einem gemeinsamen Konsens kommen können.

Die Annahme dieses Problems ist, dass es eine rote und eine blaue Armee gibt.

Die blaue Armee, bestehend aus 2 Generälen, an verschiedenen Orten, möchte sich einen Angriffszeitpunkt ausmachen.

So wird eine Nachricht mittels eines Boten von General A zu General B geschickt, wann dieser Angriff erfolgen soll. General B bekommt die Nachricht und schickt sie an A zurück. A bemerkt, dass B nicht weiß, ob die Nachricht bei A angekommen ist und so möglicherweise nicht angreift.

Deswegen schreibt A zurück an B, welcher sich bei Erhalt der Nachricht denkt, dass A nicht sicher weiß, ob die Nachricht angekommen ist.

Dieser wiederholt sich unendlich oft.
So werden lauter Nachrichten verschickt.

Es gibt bei unsicheren Verbindungen (der Bote in diesem Fall) Garantie über den Erhalt der Nachricht und somit keine letzte Nachricht, und damit auch keine Übereinkunft.

Hinweis: Bei Klausur/Prüfung diesen Sachverhalt im Rahmen eines technischen Systems erklären (zB mit Threads)

49)
Erläutern Sie die Aussage der "Byzantinischen Generäle".

Das Problem der Byzantinischen Generäle sieht folgende Situation vor:

Eine Stadt wird von einer roten Armee gehalten, während auf jedem der umliegenden Berge ein General mit seiner Armee auf den Angriff wartet.

Nun sollen die Truppen zwischen den Generälen so ausgetauscht werden, dass die Armee eines jeden Generals die gleiche Truppenstärke aufweist.

Die Generäle sind alle loyal, bis auf **EINEN**, welcher versucht die anderen so zu stören, dass es niemals zu einem Angriff kommt

Im Gegensatz zu dem TWO-ARMY Problem geht man hier davon aus, dass eine verlässliche Verbindung (Telefon) verwendet wird, immer 2 Generäle kommunizieren direkt und verlässlich miteinander und teilen sich ihre Truppenstärke mit.

Jeder General berichtet seine wahre Truppenstärke, bis auf den **fehlerhaften**, welcher jedesmal eine andere Truppenstärke liefert.

Danach hat jeder General die Informationen aller anderen. Nun teilt jeder General seine gesammelten Informationen den anderen mit, wieder lügt der falsche General und teilt jedem anderem, fehlerhafte Werte mit.

Es werden nun die Ergebnisse verglichen und die MEHRHEIT gewinnt.

Gibt es für einen Wert keine Mehrheit, so wieder dieser als UNBEKANNT maskiert.

Nun sollte jeder General die Truppenstärke der anderen wissen, nur die des falschen Generals ist unbekannt, er hat es NICHT GESCHAFFT seine Manipulation erfolgreich durchzusetzen.

Damit dieses Schema (Lamport 1982) funktioniert, benötigt man bei k fehlerhaften Prozessen mindestens 2k+1 Prozesse, welche Ordnungsgemäß arbeiten um eine Mehrheitsentscheidung treffen zu können.

Hinweis: Bei Klausur/Prüfung diesen Sachverhalt im Rahmen eines technischen Systems erklären (zB mit Threads)

50)
Erläutern Sie die Fehlerklassen in RPC-Client/server-Umgebungen. Gehen Sie besonders auf das "lost reply" Problem ein.

---

1. Der Client kann den Server nicht finden.
2. Die Anforderungsnachricht zwischen Client und Server verloren
3. Der Server stürzt ab, nachdem er eine Anforderung erhalten hat.
4. Die Antwortnachricht vom Server an den Client geht verloren
5. Der Client stürzt ab, nachdem er eine Anforderung gesendet hat.

LOST REPLY

Wenn innerhalb einer sinnvollen Zeitspanne keine Antwort eintrifft, wird die Anforderung einfach erneut gesendet.

Das Problem bei dieser Lösung besteht darin, dass der Client nicht wirklich sicher ist, warum es keine Antwort gab.

Es ist schwer zu sagen, ob die Antwort oder die Anfrage verloren gegangen ist, oder der Server in der Zwischenzeit abgestürzt ist.

Eine Möglichkeit wäre es, die Anfrage erneut zu stellen, dies ist aber nur sinnvoll bei idempotenten Operationen, das sind Operationen die wiederholt ausgeführt werden können (zB Daten lesen)

Eine Überweisung von einem Konto zu einem anderen ist **NICHT** idempotent.

Man könnte nun versuchen alle Request idempotent aufzubauen, was aber nicht immer möglich ist.

Eine weitere Möglichkeit wäre es, jeder Anfrage eine ID zu vergeben, sodass der Server erkennen kann, ob die Anfrage schon bearbeitet wurde, oder eine neue ist.

51)
Was versteht man unter reliable bzw. ordered multicast (group communication) in statischen Gruppen von Prozessen? Was muss man bedenken, wenn sich die Gruppen dynamisch verändern können? Erläutern Sie das Prinzip des "atomic multicast" ("virtual synchrony").

Was versteht man unter reliable bzw. ordered multicast (group communication) in statischen Gruppen von Prozessen?

- Solcher Dienst garantiert, dass Nachrichten an alle Mitglieder einer Prozessgruppe ausgeliefert werden.
- Kompliziertes Verfahren

Was muss man bedenken, wenn sich die Gruppen dynamisch verändern können?

?

Erläutern Sie das Prinzip des "atomic multicast" ("virtual synchrony").

ATOMIC MULTICAST

- es wird gefordert, dass alle Nachrichten in derselben Reihenfolge an alle Prozesse ausgeliefert werden.
- weiters wir die Nachricht entweder an alle oder an keinen der Prozesse ausgeliefert.
- Stellt sicher, dass nicht fehlerhafte Prozesse eine konsistente Sicht der Datenbank haben, und erzwingt einen Abgleich, wenn ein Replik wiederhergestellt wird.

VIRTUAL SYNCHRONY

Hinweis: Im Buch nachschlagen.

52)
Erläutern Sie die Definition von Security mit den Attributen Availability, Confidentiality und Integrity anhand von Beispielen. Beschreiben Sie jeweils vier Security Threats und Security Mechanisms.

Erläutern Sie die Definition von Security mit den Attributen Availability, Confidentiality und Integrity anhand von Beispielen.

AVAILIBILITY (Verfügbarkeit)

Systemstabilität
Beispiele: ?

CONFIDENTIALITY

Gibt seine Informationen nur autorisierten Parteien preis.
Beispiele: ?

INTEGRITY

Es können nur autorisierte Änderungen an den Bestandteilen des Systems vorgenommen werden.
Beispiele: ?

Beschreiben Sie jeweils vier Security Threats.

ABFANGEN (INTERCEPTION)

Bezieht sich darauf, dass eine unautorisierte Partei Zugriff auf einen Dienst oder auf Daten erlangt.

STÖREN (INTERRUPTION)

Ist die Situation, dass Dienste oder Daten zum Beispiel nicht mehr verfügbar, nicht mehr verwendbar oder zerstört sind.

Auch gilt der Begriff im Sinne von DoS-Angriffen (Denial of Service)

VERÄNDERN (MODIFICATION)

Unauthorisierte Änderungen an Daten oder Manipulationen an einem Dienst mit der Folge, dass dieser von seinen ursprünglichen Spezifikationen abweicht.

EINBRINGEN (FABRICATION)

Es werden zusätzliche Daten oder Aktivitäten erzeugt, die es normalerweise nicht gäbe. Ein Eindringling könnte zum Beispiel versuchen, einen Eintrag zu einer Passwortdatei oder einer Datenbank hinzuzufügen.

Beschreiben Sie jeweils vier Security Mechanismen.

VERSCHLÜSSELUNG

Verschlüsselung wandelt Daten in etwas um, das ein Angreifer nicht versteht.
Verschlüsselung bietet also ein Mittel zur Umsetzung von Datenvertraulichkeit.

AUTHENTIFIZIERUNG

Wird verwendet um behauptete Identität eines Benutzers, Clients oder Servers, Hosts oder einer anderen Entität zu überprüfen.

AUTORISIERUNG

Nach der Authentifizierung wird der Client überprüft, ob dieser auch autorisiert ist, die Aktion auszuführen.

KONTROLLE

Kontrollwerkzeuge werden zur Nachverfolgung verwendet, welche Clients worauf zugegriffen haben und in welcher Weise.

Auch zur Analyse von Sicherheitseinbrüchen.

53)
Was ist der Secure Socket Layer (SSL, auch TLS genannt)? Positionieren Sie SSL/TLS im Internet Protokoll Stack.

SSL steht für Secure Socket Layer und verschlüsselt eine TCP-Verbindung.

SSL befindet sich direkt über dem **TRANSPORTLAYER.**

Der Ablauf ist wie folgt:

- Client teilt dem Server seine Verschlüsselungs- und Kompressionsmöglichkeiten mit
- Der Server wählt welche davon aus und teilt dies dem Client mit
- Der Server authentifiziert sich mittels Zertifikat
- Eventuell authentifiziert sich auch der Client mittels Zertifikat
- Der Client erzeugt eine Zufallszahl, signiert sie, verschlüsselt sie mit dem Public Key des Servers und sendet sie diesem
- Beide leiten aus dieser Zufallszahl einen Session Key ab, der ab nun verwendet wird

54)
Geben Sie eine Definition für "Cryptography" an und erläutern Sie die Funktionsweise von symmetrischen und asymmetrischen Verschlüsselungsverfahren. Gehen Sie dabei auch auf die spezifischen Vor- und Nachteile ein und geben Sie konkrete Beispiele für Algorithmen an.

Geben Sie eine Definition für "Cryptography"

- Betrachten wir einen Sender S, der eine Nachricht m an einen Empfänger R senden will.
- Zum Schutz dieser Nachricht vor Sicherheitsbedrohungen VERSCHLÜSSELT der Sender sie zunächst in eine unverständliche Nachricht m‘ und sendet diese dann an R.
- R wiederum muss die empfangene Nachricht in die ursprüngliche Form m entschlüsseln
- Ver- und Entschlüsselung erfolgt über kryptographische Methoden.

Erläutern Sie die Funktionsweise von symmetrischen und asymmetrischen Verschlüsselungsverfahren. Gehen Sie dabei auch auf die spezifischen Vor- und Nachteile ein und geben Sie konkrete Beispiele für Algorithmen an.

SYMMETRISCHE VERFAHREN

- Es verwenden beide Kommunikationspartner den gleichen Schlüssel zur Ver- und Entschlüsselung.

Vorteil: viel schneller

Nachteil: sehr viele Schlüssel benötigt.

Bsp: DES

ASSYMETRISCHE VERFAHREN

- Dabei hat jeder Partner einen Schlüssel zum Verschlüsseln und einen zum Entschlüsseln.
- Der zum Verschlüsseln ist **öffentlich.**
- Der zum Entschlüsseln ist **privat.**
- Die beiden Schlüssel sind unterschiedlich.

Vorteil: nur 2n Schlüssel für n Kommunikationspartner

Nachteil: viel langsamer

55)
Welche Eigenschaften erwartet man sich von einem "Secure Channel"? Geben Sie jeweils auch Beispiele für unerwünschte Effekte an. Erklären Sie wie sich zwei Kommunikationsteilnehmer basierend auf Public Key Cryptography gegenseitig authentifizieren können.

Welche Eigenschaften erwartet man sich von einem "Secure Channel"? Geben Sie jeweils auch Beispiele für unerwünschte Effekte an.

Ein sicherer Kanal schützt Empfänger und Sender vor:

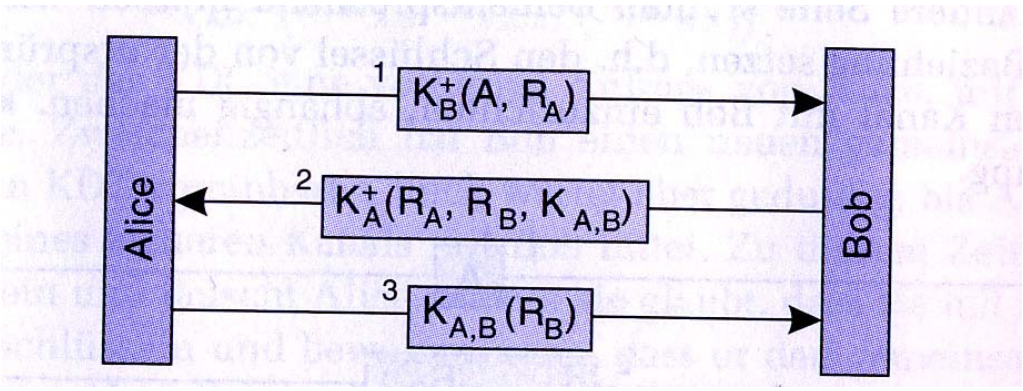
- Abfangen
- Änderung
- Fläschung

Er schützt NICHT zwangsläufig vor Störung.

Es gewährleistet, dass seine Nachrichten nicht vor Eindringlingen belauscht werden können.

Erklären Sie wie sich zwei Kommunikationsteilnehmer basierend auf Public Key Cryptography gegenseitig authentifizieren können.

- Alice sendet eine Nachricht R_A an Bob und verschlüsselt diese mit dem öffentlichen Schlüssel von Bob.
- Bob erhält diese Nachricht und entschlüsselt sie. Da Bob der einzige ist, der die Nachricht entschlüsseln kann (Bobs privater Schlüssel) weiß Alice, dass es sich um Bob handelt.
- Dieser sendet die entschlüsselte Nachricht zusammen mit der Nachricht R_B zurück (verschlüsselt mit Alice öffentlichem Schlüssel). Darüberhinaus erzeugt er einen Sitzungsschlüssel R_{AB} zur weiteren Kommunikation.
- Alice erhält die Nachricht und kann diese mit ihrem privaten Schlüssel entschlüsseln.
- Schließlich sendet Alice ihre Antwort unter Verwendung des Sitzungsschlüssels R_{AB} zurück.
- Auf diese Art beweist sie, dass sie Nachricht 2 entschlüsseln könnte und tatsächlich Alice ist, mit der Bob spricht.

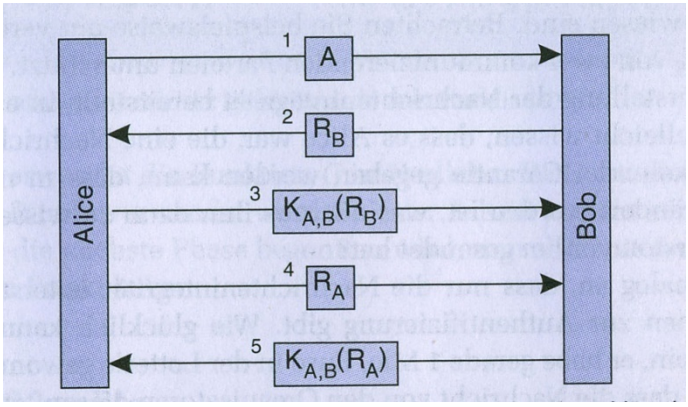


56)
Erklären Sie, wie sich zwei Kommunikationsteilnehmer basierend auf symmetrischen Schlüsseln gegenseitig authentifizieren können. Zeigen Sie auch auf, welche Probleme entstehen können, wenn ein Protokoll falsch "optimiert" wird

---

Erklären Sie, wie sich zwei Kommunikationsteilnehmer basierend auf symmetrischen Schlüsseln gegenseitig authentifizieren können.

- Zuerst sendet Alice Ihre Identität (Nachricht 1), um zu signalisieren, dass sie einen Kommunikationskanal einrichten möchte.
- Dann sendet Bob Nachricht R_B an Alice.
- Alice empfängt diese Nachricht verschlüsselt sie und sendet sie verschlüsselt zurück an Bob.
- Dieser entschlüsselt sie und wenn der Inhalt mit dem gesendeten übereinstimmt weiß er, dass es sich um Alice handelt.
- Nun wieder dasselbe von Alice-Seite aus erledigt. Alice sendet die Nachricht R_A an Bob.
- Bob empfängt diese Nachricht, verschlüsselt sie und sendet sie zurück an Alice.
- Alice empfängt diese Nachricht, prüft ob der Inhalt mit dem gesendeten übereinstimmt -> wenn ja weiß sie dass sie mit Bob kommuniziert.



Zeigen Sie auch auf, welche Probleme entstehen können, wenn ein Protokoll falsch "optimiert" wird

Wenn das oben beschrieben Protokoll in nur 3 Schritten statt 5 durchgeführt wird, kann es zu einem **REFLEKTIONSANGRIFF** kommen, sodass ein Eindringlich (zB Chuck) einen Kanal mit Bob einrichten kann und dieser im vortäuscht, dass er mit Alice redet.

57)
Was ist eine digitale Signatur, welche Garantien bietet sie und wie funktioniert sie? Was ist eine Hash Funktion und welche Eigenschaften muss sie erfüllen, damit sie im Rahmen einer digitalen Signatur eingesetzt werden kann.

Was ist eine digitale Signatur, welche Garantien bietet sie und wie funktioniert sie?

- Eine digitale Signatur ist ein kryptografisches Verfahren, bei dem zu einer „Nachricht“ (dh. zu beliebigen Daten) eine Zahl (die digitale Signatur) berechnet wird, deren Urheberschaft und Zugehörigkeit zur Nachricht durch jeden geprüft werden können.
- Digitale Signaturen basieren auf asymmetrischen Kryptosystemen und verwenden folglich ein Schlüsselpaar, das aus einem privaten (geheimen) und einem öffentlichen (nicht geheimen) Schlüssel besteht.
- Diese Schlüssel werden im deutschen Signaturgesetz als (privater) Signaturschlüssel und (öffentlicher) Signaturprüfschlüssel bezeichnet.
- Oftmals findet man dafür die englischen Bezeichnungen Private Key und Public Key.

Bietet Garantie über die

- Urheberschaft
- Nachricht kann vom Empfänger nicht abgeändert werden
- Das Senden der Nachricht kann nicht bestritten werden.

Was ist eine Hash Funktion und welche Eigenschaften muss sie erfüllen, damit sie im Rahmen einer digitalen Signatur eingesetzt werden kann.

Eine Hashfunktion H erzeugt aus einer Nachricht m von beliebiger Länge eine Bit-Folge h mit fester Länge.

h = H(m)

zB MD5 ist eine Hash-Funktion zur Berechnung eines Message Digest fester Länge von 128 Bits aus einer beliebigen langen binären Zeichenkette.

EIGENSCHAFTEN

- Ist Einwegfunktion, dh es ist nicht möglich über Berechnungen die Eingabe herauszufinden
- Schwache Kontrollresistenz
  es ist praktisch unmöglich sein, zwei Nachrichten zu finden, deren Hashwert identisch ist
- Starke Kontrollresistenz
  zwei unterschiedliche Eingaben produzieren immer unterschiedliche Hashwerte (niemals gleiche)

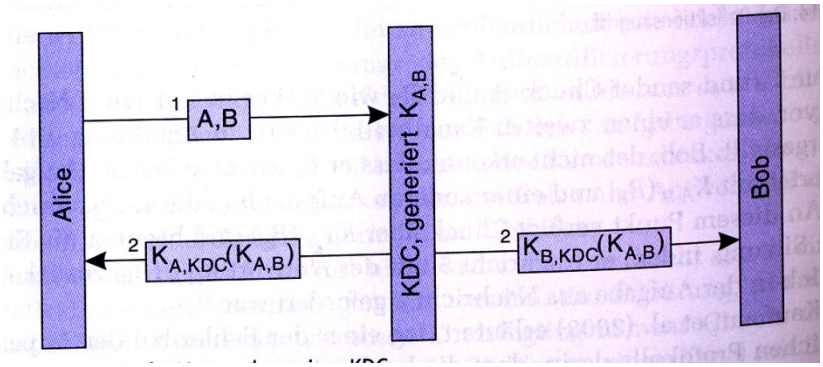
58)
Was ist ein Key Distribution Centre (KDC), wozu wird es eingesetzt und welchen Vorteil bietet es? Geben Sie ein konkretes Verfahren zur gegenseitigen Authentifizierung von Kommunikationsteilnehmern mit Hilfe eines KDC an und beschreiben Sie für jeden Schritt, wer sich wem gegenüber bereits authentifiziert hat.

Was ist ein Key Distribution Centre (KDC), wozu wird es eingesetzt und welchen Vorteil bietet es?

- Eines der Probleme bei der Verwendung eines gemeinsamen geheimen Schlüssels zur Authentifizierung ist die Skalierbarkeit.
- Wenn ein verteiltes System N Hosts enthält und jeder Host gemeinsam mit jedem der anderen N-1 Hosts einen geheimen Schlüssel nutzen muss, erfordert dies, dass das System als Ganzes N (N-1) / 2 und jeder Host N-1 Schlüssel verwalten muss.
- Bei großen N führt dies zu Problemen.
- Ein Alternativer Ansatz ist die Verwendung eines KDC, das die Schlüssel verwaltet.

Geben Sie ein konkretes Verfahren zur gegenseitigen Authentifizierung von Kommunikationsteilnehmern mit Hilfe eines KDC an und beschreiben Sie für jeden Schritt, wer sich wem gegenüber bereits authentifiziert hat.

- Die Idee ist, dass das KDC sowohl einen Schlüssel an Alice als auch an Bob ausgibt, den sie dann zur Kommunikation nutzen können.
- Alice sendet zunächst eine Nachricht an das KDC mit der Mitteilung, dass sie mit Bob reden will
- Das KDC sendet eine Nachricht zurück, die einen gemeinsamen geheimen Schlüssel $K_{A,B}$ enthält, den sie verwenden kann.
- Die Nachricht wird mit dem geheimen Schlüssel $K_{A,KDC}$ den es mit Alice gemeinsam nutzt verschlüsselt.
- Darüberhinaus sendet das KDC $K_{A,B}$ auch an Bob, nun aber verschlüsselt mit dem geheimen Schlüssel $K_{B,KDC}$, den es mit Bob gemeinsam nutzt.



59)
Ordnen Sie in der folgenden Taxonomie den vier Typen von Koordinationsmodellen jeweils zwei konkrete Techniken, Beispiele oder Systeme zu.

		Temporal	
		Coupled	Uncoupled
Referential	Coupled	(a) Direct	(b) Mailbox
	Uncoupled	(c) Meeting oriented	(d) Generative Communication

a)
1. ICQ
2.

b)
1. E-Mail
2.

c)
1. Publish/Subscribe Systeme
2.

d)
1. Jini
2.

60)
Erläutern Sie das Grundprinzip von publish/subscribe als Kommunikations-/Koordinationsmechanismus. Worin bestehen die Möglichkeiten aber auch die Probleme dieses Mechanismus (v.a. dessen Implementierung). Erläutern Sie weiters JINI und JavaSpaces als konkrete Technologien.

Erläutern Sie das Grundprinzip von publish/subscribe als Kommunikations-/Koordinationsmechanismus. Worin bestehen die Möglichkeiten aber auch die Probleme dieses Mechanismus (v.a. dessen Implementierung).

- Publish/Subscribe-Systeme, bei denen Nachrichten mit einem gewissen Typ versehen werden.
- Diese werden an alle Prozesse zugestellt, die sich dafür interessieren (indem sie sich vorher dafür registriert haben).
- Funktioniert nur wenn beide Prozesse aktive sind.
- Dieser Ansatz wird als Meeting-based bezeichnet.

Erläutern Sie weiters JINI und JavaSpaces als konkrete Technologien.

Jini setzt Generative Communication um.

- Es speichert Tupel von Java-Objekten in einem Shared Dataspace (JavaSpace), aus dem sie von interessierten Prozessen durch Matching mit einem Template (ebenfalls ein Tupel aus Objekten, das aber nicht vollständig ausgefüllt sein muss) wieder abgefragt werden können.
- Das Abfragen kann dabei auch das gelesene Tupel auch gleich aus dem Datastore entfernen (take-Operation).
- Neben dem Zugriff auf JavaSpaces unterstützt Jini auch noch Naming und Security (Authentification, so dass nur berechtigte Prozesse auf den Datastore zugreifenkönnen)

61)
Was ist ein Web-Server? Wozu dient das Hypertext Transfer Protokoll HTTP? Nennen Sie zwei HTTP Operationen. Diskutieren Sie die Funktionsweise eines CGI-Programms.

Was ist ein Web-Server?

Ein Webser ist ein Programm das eingehende HTTP Anforderungen erfüllt, indem er das angeforderte Dokument lädt und es an den Client verschickt.

Wozu dient das Hypertext Transfer Protokoll HTTP?

- HTTP ist ein allgemeines Client-Server-Protokoll, das für viele dokumentenbasierte Übertragungen im Internet verwendet wird.
- Es ist nicht für eine bestimmte Anwendung spezialisiert.
- Am bekanntesten ist die Übertragung gewöhnlicher Webseiten, es können aber auch ganze Anwendungen auf http aufgebaut werden.
- Obwohl http selbst zustandslos ist kann man mit Hilfe von Cookies ein zustandsbehaftetes Protokoll simulieren, was für moderne Webabwendungen notwendig ist.
- Insgesamt wird http verwendet um von Clients aus bestimmte Services in Anspruch nehmen zu können.

Nennen Sie zwei HTTP Operationen

GET
Dokument an den Client schicken

PUT
Dokument speichern

POST
Bereitstellung von Daten die einem Dokument hinzugefügt
werden soll

DELETE
Dokument löschen

Head Anforderungen, Head eines Dokument zurückzugeben

Diskutieren Sie die Funktionsweise eines CGI-Programms.

- CGI – Common Gateway Interface
- CGI deffiniert eine Standardmethode, wie der Webserver ein Programm ausführen kann und die Benutzereingaben verwenden kann.
- CGI Programme arbeiten meist mit der lokal auf dem WEB server gespeicheten Datenbank.
- Nach Verarbeitung der Daten erzeugt das Programm ein HTML-Dokument und gibt dies dem Server zurück, der es dem Client weiter gibt. (Generierte Dokumente)